UNIVERSITÀ DI PISA

Facoltà di Scienze, Matematiche, Fisiche e Naturali

Corso di laurea in Informatica

Progettazione e Sviluppo di un Ambiente Grafico per Mashup di Applicazioni Web

Candidato:

Giuliano Pintori

Relatore:

Prof. Fabio Paternò

Anno accademico 2011/2012

Sommario

Lo scopo di questa tesi è la realizzazione un ambiente grafico a manipolazione diretta per creare mashup di applicazioni Web. La sua caratteristica principale, è quella di rendere più facile e flessibile lo sviluppo di tali applicazioni anche ad utenti che non hanno esperienza di programmazione. Nella piattaforma è stata anche introdotta la possibilità di creare comunità di utenti che possono condividere, commentare, valutare le loro soluzioni, anche tramite il social network Facebook.

Abstract

The purpose of this thesis is to create a graphical direct manipulation environment to create Web mashups. Its main characteristic is to make easier and more flexible the development of such applications to users who have no programming experience. The platform also introduces the ability to create communities of users that can share, comment, rate their solutions, even using the social network Facebook.

Indice

1 Introduzione **11**

2 Stato dell'arte **16**

2.1 End User Development e personalizzazione del Web 16

2.1.1 La personalizzazione del WEB 17

2.2 Mashups 21

2.3 Tool che permettono di personalizzare il WEB 24

2.3.1 Greasemonkey 24

2.3.2 Chickenfoot 25

2.3.3 CoScripter 26

2.3.4 Clip, Connect, Clone 28

2.3.5 Intel Mash Maker 29

2.3.6 iGoogle 30

2.3.7 Yahoo! Pipes 31

2.3.8 WSO2 Mashup Server 32

2.3.9 Microsoft Popfly 33

2.3.10 EzWeb Enterprise Mashup 34

2.3.11 d.mix 34

2.3.12 Crowdsourcing Platform and CrowdDesign 34

2.3.13 Confronto tra gli strumenti analizzati 36

3 Background **39**

3.1 Architettura della piattaforma 39
3.2 Selezione delle componenti 41
3.3 Riconoscimento dei parametri di input 46
3.4 Connessione tra i widget 47
3.5 Aggiornamento del mashup 49
3.6 Limitazioni della versione precedente 51

4 Ambiente EUD[1] per mashup di applicazioni Web **52**

4.1 Introduzione 52
4.2 Modifiche alla creazione dei widget 53
4.3 Modifiche alla gestione delle connessioni 55
4.4 Sistema di visualizzazione delle connessioni 60
4.5 Gestione degli utenti 62
4.6 Salvataggio e caricamento dei mashup dell'utente 63
4.6.1 Libreria dei widget 63
4.6.2 Gestione dei mashup 69
4.7 Implementazione 73

5 Supporto alla creazione di una comunità di utenti **80**

5.1 Implementazione del repository 80
5.2 Condivisione di un mashup 82
5.3 Caricamento di un mashup 84
5.4 Votazione e Commenti 86
5.5 Condivisione di un widget 89
5.6 Caricamento di un widget 91
5.7 Implementazione 93

[1]End User Development

6 Integrazione con Facebook **96**
6.1 Facebook 97
6.2 Facebook Platform 98
6.3 MashupEditor Application 102
6.4 Login tramite Facebook 104
6.5 Share e Like Button 107
6.6 Comments 113
6.7 Open Graph Protocol 117

7 Un esempio di applicazione **125**
7.1 Scenario 125
7.2 Parte 1: Creazione e condivisione di un mashup 126
7.3 Parte 2: Caricamento e modifica di mashup 135

8 Validazione **139**
8.1 Test di valutazione dell'usabilità 139
8.1.1 Organizzazione del test 139
8.1.2 Partecipanti 140
8.1.3 Scenario e struttura del test 141
8.2 Analisi dei risultati del test per i task proposti 145
8.2.1 Task 1 145
8.2.2 Task 2 148
8.2.3 Task 3 149
8.2.4 Task 4 151
8.3 Considerazioni finali sui risultati del test di usabilità 154

9 Conclusioni **156**

A Esempio di file di salvataggio XML di un mashup **159**

A.1 Schema XSD per il salvataggio dei mashup 159

A.2 Esempio di file XML di salvataggio di un mashup 163

B Questionario Test Utente **169**

B.1 Informazioni personali 169

B.2 Esperienze dell'utente con l'uso di strumenti per il mashup . . 170

B.3 Domande sul Task 1 170

B.4 Domande sul Task 2 170

B.5 Domande sul Task 3 171

B.6 Domande sul Task 4 172

B.7 Domande finali 172

Bibliografia **173**

Elenco delle figure

1.1 Struttura della tesi 14

2.1 Mashup su Google Maps 21

3.1 Struttura della piattaforma di mashup 40
3.2 Azioni che vengono eseguite per la creazione di un widget . . . 42
3.3 Navigazione verso il sito Web 43
3.4 Selezione delle parti di applicazione 44
3.5 Inserimento del titolo del nuovo widget 44
3.6 Visualizzazione del nuovo widget all'interno del MashupEditor 45
3.7 Schermata di riconoscimento dei parametri 46
3.8 Schermata della finestra per la connessione tra widget 48
3.9 Risultato finale della creazione di un mashup 50
3.10 Aggiornamento del mashup 50

4.1 Aspetto del widget nella vecchia versione 54
4.2 Aspetto del widget nella nuova versione 54
4.3 Attivazione della modalità di registrazione 56
4.4 Schermata dell'editor in modalità registrazione 57
4.5 Copia del contenuto del text field del widget sorgente 57
4.6 Il contenuto degli appunti viene incollato all'interno del text field del widget destinazione 58
4.7 Stop alla registrazione 58
4.8 La piattaforma mostra il solo il text field del widget sorgente . 58

4.9 Contenuto del mashup prima dell'esecuzione 59
4.10 Contenuto del mashup dopo l'esecuzione 59
4.11 Attivazione della modalità di visualizzazione delle connessioni 60
4.12 Connessioni presenti all'interno del mashup 61
4.13 Stop della modalità di visualizzazione 61
4.14 Aggiornamento del contenuto del mashup 62
4.15 Nuova veste grafica del MashupEditor 63
4.16 Avvio il salvataggio del widget nella libreria 65
4.17 Finestra di dialog dove inserire le informazioni relative al widget 66
4.18 Informazioni riguardanti il widget 67
4.19 Comando per il caricamento del widget dalla libreria 67
4.20 Elenco dei widget presenti nella libreria 68
4.21 Il nuovo widget viene inserito all'interno dell'ambiente di composizione 68
4.22 Elenco dei comandi per la gestione di un mashup 70
4.23 Finestra per la creazione di un nuovo progetto mashup 71
4.24 Inserimento delle informazioni sul nuovo progetto 72
4.25 Schermata che mostra i progetti dell'utente 72

5.1 Struttura del database del repository 81
5.2 Comandi del menù Share 82
5.3 Finestra per l'inserimento delle informazioni del mashup da condividere 82
5.4 Esempio di URL univoca nel sistema 83
5.5 Finestra per la ricerca nel repository 84
5.6 Scelta della tipologia di ricerca 84
5.7 Selezione del mashup desiderato 85
5.8 Caricamento del mashup selezionato 85
5.9 Visualizzazione delle funzioni voto e commento 86
5.10 Finestra con i commenti precedenti 87
5.11 Inserimento di un commento 1 87
5.12 Inserimento di un commento 2 88

5.13 Visualizzazione del rating del mashup 88
5.14 Inserimento di un voto 89
5.15 Rating del mashup aggiornato dopo l'inserimento del voto . . 89
5.16 Comando per la condivisione di un widget nel repository . . . 90
5.17 Finestra per l'inserimento delle informazioni relative al widget 90
5.18 Comando per il caricamento di un widget dal repository . . . 91
5.19 Risultati della ricerca di un widget 92
5.20 Il nuovo widget all'interno dell'editor 92

6.1 Schermata per la creazione dell'applicazione MashupEditor . . 102
6.2 Pannello di controllo dell'applicazione MashupEditor 103
6.3 Home Page del MashupEditor che mostra il Login Button . . . 106
6.4 Finestra dove si inseriscono le credenziali per l'autenticazione in Facebook 106
6.5 Menù principale dell'applicazione che visualizza il tasto Logout da Facebook 106
6.6 Schermata dell'editor con un mashup aperto 107
6.7 Schermata per la condivisione del mashup su Facebook 109
6.8 Box del mashup condiviso all'interno della bacheca di Facebook 109
6.9 Esprimo la preferenza tramite il Like Button 111
6.10 Box generato dall'azione like all'interno della bacheca di Facebook 111
6.11 Utilizzo del Send Button 111
6.12 L'utente destinatario del messaggio riceve il link al mashup . . 112
6.13 Schermata dell'editor che contiene il plugin Comments 114
6.14 Inserimento di un nuovo commento 115
6.15 Il commento pubblicato appare su Facebook 115
6.16 Inserimento di un commento ad un commento 115
6.17 Il plugin aggiorna il suo contenuto con il nuovo commento inserito su Facebook 116
6.18 Definizione dell'Action "Create" 1 117
6.19 Definizione dell'Action "Create" 2 118

6.20 Definizione dell' Object "Mashup" 119
6.21 Schema iterazioni con il protocollo Open Graph 120
6.22 Finestra inserimento informazioni sul mashup da condividere . 121
6.23 Notizia della creazione di un mashup sulla bacheca di un amico 122
6.24 Notizia della creazione di un mashup sulla timeline dell'utente 122
6.25 Elenco dei mashup creati dall'utente 123
6.26 Riassunto dei mashup creati dall'utente per periodo temporale 123
6.27 Elenco dei mashup creati dall'utente durante il mese di giugno 123

7.1 Creazione del progetto 126
7.2 Creazione di un nuovo widget tramite la navigazione Web . . . 127
7.3 Selezione della form di ricerca sul sito di Amazon 127
7.4 Selezione dei risultati della ricerca effettuata su Amazon . . . 128
7.5 Inserimento del nome del widget 128
7.6 Il nuovo widget all'interno dell'ambiente grafico 129
7.7 Finestra per la ricerca dei widget nel repository 129
7.8 I due widget all'interno dell'ambiente grafico 130
7.9 Operazione copia 131
7.10 Operazione incolla 131
7.11 Comando di stop della registrazione 132
7.12 Aspetto dei due widget dopo la connessione 132
7.13 Risultato dell'esecuzione del mashup 132
7.14 Finestra per l'inserimento delle informazioni del mashup da condividere 133
7.15 Le funzioni di voto e commento vengono visualizzate all'interno del'editor 134
7.16 Notizia sulla creazione del mashup all'interno della bacheca dell'utente 134
7.17 Link sulla bacheca di Facebook 135
7.18 L'ambiente grafico mostra il mashup caricato 135
7.19 Caricamento del widget di wikipedia dalla libreria 136

7.20 Il widget caricato dalla libreria viene inserito all'interno del mashup 136
7.21 Esecuzione del mashup 137
7.22 Condivisione delle modifiche nel repository 138
7.23 Alessandra invia un messaggio a Giuliano per informarlo della modifiche effettuate 138
7.24 Giuliano riceve il messaggio da Alessandra 138

8.1 Social Network utilizzati dagli utenti del test 140

Elenco delle tabelle

2.1 Confronto tra gli strumenti 37

Capitolo 1

Introduzione

Il Web è ancora l'interfaccia utente più comune. Milioni di siti Web sono disponibili, e sempre più persone vi accedono quotidianamente per le loro attività.
Tuttavia, ogni utente, ha differenti esigenze e attività da svolgere. Questo ha aumentato l'interesse per i *Mash-up*, il cui scopo è quello di comporre interfacce utente, contenuti e funzionalità, provenienti da varie fonti.
I mashup sono applicazioni Web, generate dalla combinazione di contenuti, presentazioni o funzionalità di diverse applicazioni o sorgenti di dati presenti nel Web. I mashup mirano a combinare queste fonti per creare nuove applicazioni utili.
I primi studi sugli sviluppatori web e le loro esperienze con la costruzione di mashup, hanno dimostrato che la maggior parte di essi creavano mashup che avevano come base le mappe geografiche, mentre altri creavano mashup utilizzando applicazioni per la gestione di foto come *Flickr*.
Questo è cambiato nel tempo, e ambienti come *iGoogle*, hanno dimostrato che la tecnica del mashup può essere utilizzata per costruire un'ampia varietà di applicazioni composte.
Diversi studi hanno anche evidenziato una serie di problemi con ambienti mashup: spesso, l'utente utilizzatore, doveva possedere dei concetti di base di programmazione per poter utilizzare le applicazioni per la creazione di

mashup.
Il nostro obiettivo è superare tale limite tramite un ambiente grafico per l'EUD (*End User Development*) di applicazioni mashup, cioè un ambiente che consenta all'utente finale che, non necessariamente, possiede concetti di programmazione anche basilari, di creare, sviluppare e comporre delle applicazioni web.

Obiettivi

Il mio lavoro di tesi si colloca all'interno dell'attività di ricerca svolta presso il laboratorio HIIS[1] dell'ISTI[2] al CNR[3] di Pisa. Nello specifico, mi sono occupato della progettazione e implementazione di nuove funzionalià per un ambiente grafico per l'EUD di Mashup di applicazioni Web.

Gli obiettivi di questa tesi sono:

1. progettare e implementare le modifiche necessarie per accrescere le funzionalità di EUD della piattaforma;
2. potenziare il supporto per della piattaforma verso gli utenti consentendo ad essi di creare, salvare e ricaricare mashup di applicazioni Web;
3. inserire aspetti sociali e collaborativi all'interno dell'ambiente di sviluppo dando la possibilità agli utenti di condividere i propri mashup con gli altri utenti della piattaforma;
4. integrare il mashup editor con il social network Facebook tramite i plugin e le funzionalità che sono messi a disposizione da Facebook;
5. mostrare i risultati ottenuti proponendo un esempio di creazione di un mashup;
6. valutare e validare le modifiche introdotte attraverso un test utente.

[1]Human Interfaces in Information Systems
[2]Istituto di Scienza e Tecnologie dell'Informazione
[3]Consiglio Nazionale della Ricerca

Contenuto

Il contenuto di questa tesi è strutturato secondo la figura 1.1.

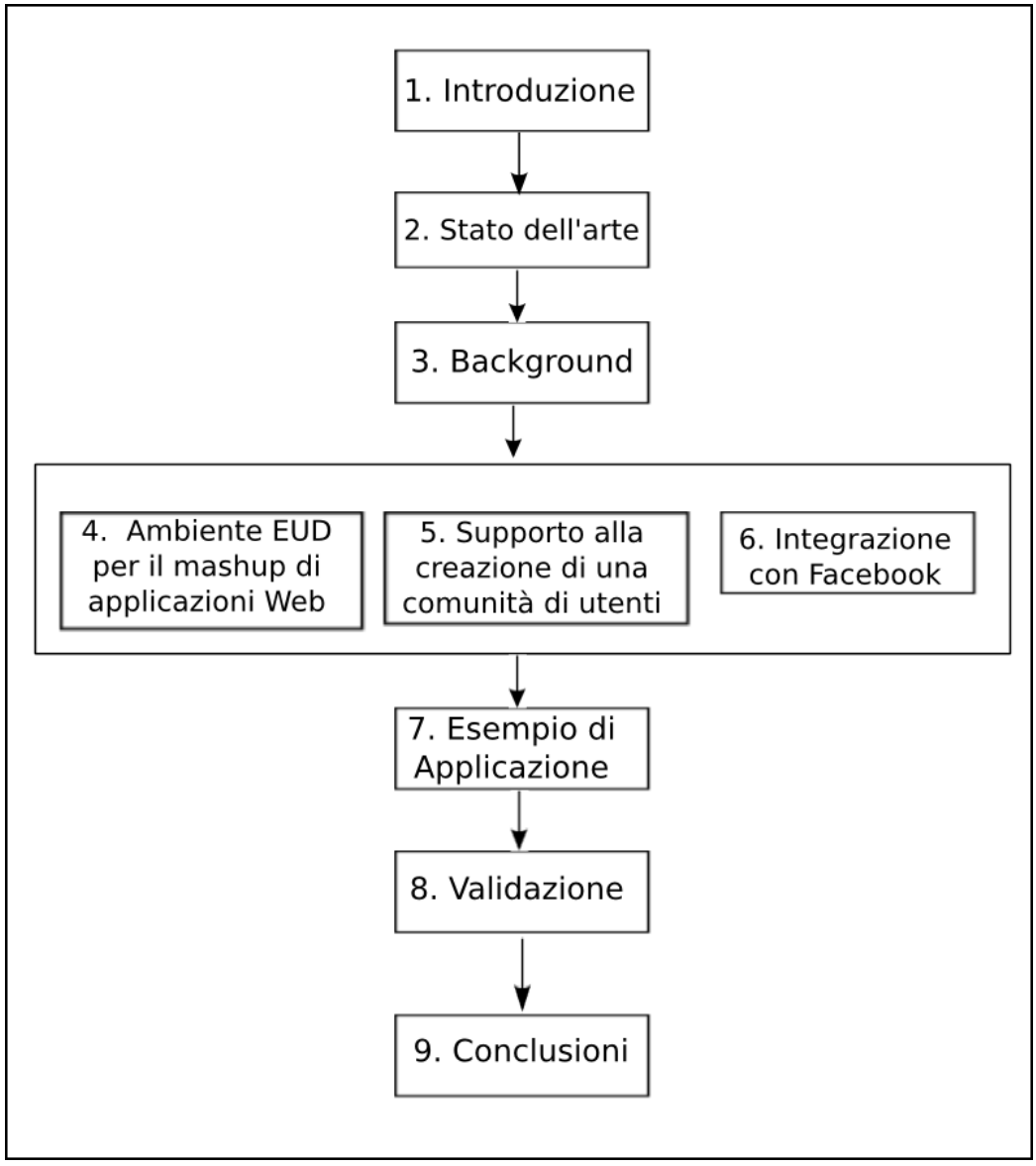

Figura 1.1: Struttura della tesi

Il capitolo 1 introduce una breve descrizione del lavoro di tesi, evidenziandone i principali obiettivi.

Il capitolo 2 si dedica allo stato dell'arte, introducendo i principi che hanno ispirato la creazione di questo strumento di sviluppo. Successivamente, viene fatta una panoramica dei vari tool che supportano tipologie di utilizzo simili a quella da noi proposta.

Il capitolo 3 presenta lo stato e le funzionalità del MashupEditor che ho trovato all'inizio del mio percorso di tesi.

I capitoli 4, 5 e 6 mostrano il mio lavoro di progettazione e sviluppo durante il periodo di tesi, in particolare:

- Il capitolo 4 è incentrato sulla prima parte del lavoro di tesi, vengono presentate le modifiche da me studiate per aggiungere funzionalitá alla piattaforma e trasformarla in un EUD environment;
- Il capitolo 5 si concentra sull'implementazione di una *Community* tramite la quale gli utenti della piattaforma possono condividere, valutare e commentare le loro soluzioni;
- Il capitolo 6 presenta le modifiche introdotte per integrare il MashupEditor con l'ambiente dei social network, mostrando l'integrazione con la piattaforma Facebook.

Il capitolo 7 mostra un caso d'uso per la piattaforma dopo le modifiche introdotte.

Il capitolo 8 presenta il test utente realizzato per valutare le modifiche introdotte, mostrando i risultati ottenuti.

Il capitolo 9 riassume il lavoro di tesi, analizzando i risultati ottenuti e proponendo nuove idee per futuri sviluppi.

Capitolo 2

Stato dell'arte

2.1 End User Development e personalizzazione del Web

L'*End User Development* [8] é un approccio alla programmazione che consente agli utenti che non sono dei programmatori di poter sviluppare le proprie applicazioni.
Negli anni '60, usare un computer significava programmare un computer, non esisteva dunque il concetto di "*end user programming*" poichè tutti gli utenti erano programmatori.
Negli anni '80 l'introduzione del Macintosh ha portato un cambiamento importante, ci si riferiva ai computer definendoli "computer desktop". Le tipologie di programmazione si sono adattate e la programmazione da riga di comando ha via via lasciato spazio alla programmazione delle interfacce utente, è così che venne a crearsi il concetto di "*end user*".
Gli end users non volevano essere programmatori, l'uso principale che facevano del computer era la scrittura di documenti, la creazione di grafici e fogli di calcolo, cioè quella che oggi viene ritenuta la base per l'uso di un computer.
Naturalmente nel tempo sono emerse delle esigenze da parte dell'utente finale, come ad esempio, automatizzare sequenze di operazioni da ripetere nel tempo, oppure rendere automatiche operazioni che richiedono un imprecisa-

to numero di azioni per essere completate. In quest'ottica, l'utente potrebbe aver bisogno di semplificare le proprie interazioni con il computer.
Viene così a crearsi la necessità di aiutare l'utente finale a costruire le proprie applicazioni senza che egli abbia necessariamente preconcetti di programmazione.
Il termine "*end user*" si riferisce alla maggior parte degli utenti che utilizzano i computer per sfruttare i software disponibili, la portata delle azioni che possono compiere rimane quindi limitata dalle caratteristiche di tali software. La parola "*end*" distingue questo gruppo di utenti dai programmatori che utilizzano i computer per creare tool e applicazioni di cui gli altri utenti poi usufruiscono.
La sfida principale per i ricercatori è trovare delle forme di programmazione che risultino facilmente comprensibili e piacevoli da utilizzare, e far si che gli *end user* siano così disposti e capaci a creare nuove applicazioni.
Il Web moderno si è affermato come concorrente principale degli strumenti desktop, offrendo agli utenti non solo notizie e media, ma anche la possibilità di utilizzare delle applicazioni. Le potenzialità della piattaforma Web si sono rivelate tali che sempre più applicazioni sono state sviluppate per essere utilizzate on-line (pensiamo a siti per l'E-commerce, social network, siti per la gestione delle caselle di posta) e ogni utente può sentire la necessità di personalizzare i contenuti di tali applicazioni per adattarli alle proprie preferenze.

2.1.1 La personalizzazione del WEB

La flessibilità della piattaforma Web viene in aiuto degli utenti, rendendo possibile delle personalizzazioni che non sarebbero possibili sui programmi per il desktop, questo perchè le pagine Web utilizzano HTML[1] per visualizzare le interfacce utente, e la rappresentazione dell'HTML tramite DOM[2] all'interno dei browser rende possibile l'ispezione e la modifica delle pagine

[1]HyperText Markup Language
[2]Document Object Model

stesse. È possibile anche intervenire sugli eventi che vengono scatenati dall'utente all'interno delle pagine, come ad esempio quando egli effettua dei click o invia delle form, e personalizzare il comportamento della pagina Web quando vengono intercettati determinati eventi.
Inoltre le personalizzazioni possono essere inserite all'interno delle pagine senza dover passare necessariamente attraverso gli sviluppatori del sito, al contrario per modificare i programmi desktop bisogna passare attraverso il team di sviluppatori del programma, che devono fornire esplicitamente gli eventuali strumenti per aggiungere o modificare le personalizzazioni.

Gli aspetti principali sui quali si concentrano le personalizzazioni sono [11]:

- *Il tipo di sito che si vuole personalizzare*:

  La principale tipologia di siti Web che vengono customizzati sono quelli per la gestione di informazioni personali come email, calendars, to-do lists, generalmente siti dove l'utente passa molto tempo. Altri tipi di siti customizzabiti sono quelli di foto e video-sharing.

- *La natura della personalizzazione*:

  - *Shortcuts*: si automatizza e si ottimizza l'esecuzione di task frequenti o che vengono ripetuti spesso come la gestione delle caselle di posta elettronica oppure il riempimento automatico di form;
  - *Semplification*: si tende a ridurre il disordine all'interno delle pagine e le probabili distrazioni dell'utente oppure ad eliminare delle features non necessarie. Ad esempio si eliminano pubblicità e oggetti Flash dalle pagine per allegerire l'esecuzione delle pagine;
  - *Mashups*: si combinano le infomazioni tra due o più siti diversi. L'uso tipico consiste nell'accrescere la quantità di informazione utile per l'utente, integrando quella presente su un sito con informazioni (anche di tipo diverso) prelevate da altri siti;

- *Personalizzazioni generiche vs. versioni specializzate dei siti*:

  Si costruiscono personalizzazioni che sono dirette a specifici siti come *Gmail, Youtube, Flickr, Wikipedia,* oppure personalizzazioni che vengono utilizzate in più siti come ad esempio l'anteprima del contenuto degli hyperlink che visualizza un pop up al passaggio del mouse, oppure il vocabolario o la traduzione di parole selezionate col mouse.

- *Personalizzazioni per task singoli o task che devo essere ripetuti*:

  Gli esempi visti nel punto precedente automatizzano dei task che vengono ripetuti più volte dal singolo utente e che avrebbero poco valore se utilizzati poche volte; invece si parla di task singoli quando si automatizzano delle procedure particolarmente lunghe che non devono essere ripetute di frequente come ad esempio il calcolo automatico dei punteggi di una gara, la selezione automatica di tutte le persone in una mailing-list, etc.

- *Chi crea le personalizzazioni*:

  La maggior parte delle personalizzazioni vengono create dagli utenti dei siti, la maggior parte dei quali ha però conoscenze di programmazione superiore alla media. A volte sono i siti stessi che costruiscono le personalizzazioni spesso in forma di extensions per i vari browser, ad esempio le varie *Google Toolbar, Yahoo Toolabar*, etc. Ci sono poi i siti open-source il cui codice è aperto e modificabile anche dagli utenti come in *Mailman, MediaWiki* e *IBM CoScripter*. Infine possiamo trovare dei ricercatori che non sono ne utenti finali ne sviluppatori che esplorano le interfacce web per sviluppare dei prototipi di ricerca.

- *La relazione tra la personalizzazioni e il sito target*:

  Alcune personalizzazioni potrebbero infrangere le regole di un sito, come ad esempio script che abilitano link, che fanno sparire le pubblicità che il sito supporta oppure script che automaticamente consentono all'utente di giocare on-line, di completare form, ecc..

Ci sono siti che si difendono dall'automatizzazione introducendo ad esempio il "*CAPTCHA*" per distinguere l'utente umano da uno simulato da uno script.
Infine ci sono stati dei casi in cui le personalizzazioni hanno portato ad un miglioramento dell'usabilità di un sito, come nel caso di Gmail dove l'estensione "*Gmail Delete Button*" è diventata obsoleta poichè ora Gmail include il pulsante "*delete*".

2.2 Mashups

Con il termine Mashup si indicano le applicazioni Web ottenute dalla combinazione di contenuti, presentazioni o funzionalità di diverse applicazioni o sorgenti di dati nel Web, all'interno di un unica interfaccia. Le prime applicazioni sviluppate con questi criteri sono state quelle basate sulle mappe: visualizzare i punti di interesse di un luogo visitato (monumenti, ristoranti, negozi, ecc.) all'interno di una mappa viene considerato creare un Mashup. All'interno di Google Maps ora troviamo diversi Mashup: è possibile visualizzare all'interno delle mappe diverse informazioni provenienti da diverse fonti, nella figura 2.1 viene mostrata l'integrazione tra Google Maps, Wikipedia e Panoramio.

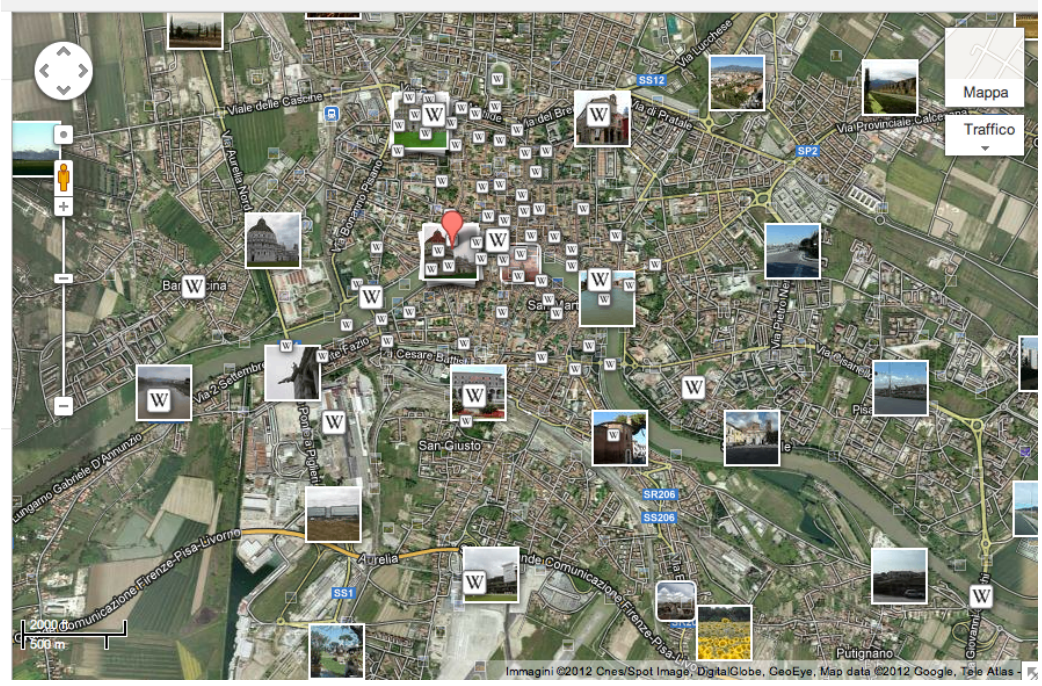

Figura 2.1: Mashup su Google Maps

Possiamo individuare due categorie di Mashups:

1. *Enterprise mashups*: Gli enterprise mashups si riferiscono ad applicazioni che combinano risorse, applicazioni e dati eterogenei provenienti da varie sorgenti per risolvere problematiche aziendali. Ci si riferisce a tali mashup anche come *Business mashup* oppure come *Data mashup*. Il target di questa categoria di mashup sono le aziende a differenza dei *Web Mashup* che sono destinati a tutti gli utenti della rete.
   Un *Enterprise mashup* è la combinazione di applicazioni e dati interni all'aziende con sorgenti esterne di dati, SaaS[3] e contenuti Web. I mashup di questa categoria sono integrati con il business computing environment, la data governance, con la BI[4]/BA[5] presenti all'interno di un'azienda.
   Gli enterprise mashup si presentano come delle applicazioni Web dalle quali è possibile visualizzare interattivamente le informazioni prese dalle diverse fonti, sia interne che esterne.
   I mashup di questa categoria possono essere utilizzati per integrare le ricerche di mercato delle aziende, ad esempio un'azienda che vende merchandising sportivo attraverso la propria BI scopre che in passato, le vendite di magliette raddoppiano dopo tre vittorie consecutive in casa per la squadra locale.
   Per massimizzare il profitto bisognerebbe avere a disposizione tutte le magliette nel negozio durante la partita giusta. Per individuare quando portare la giusta quantità di magliette l'azienda dovrebbe analizzare i risultati della squadra, invece di memorizzare i dati relativi alle partite della squadra, è possibile prelevarli da una fonte esterna e combinarli con i dati di vendita dell'azienda per individuare i giorni in cui verranno vendute più magliette;

[3]Software as a Service
[4]Business Intelligence
[5]Business Analytics

2. *Consumer mashups*: rappresentano la categoria di mashup destinata a tutti gli utenti della rete. I consumer mashup rappresentano un tipico esempio del Web 2.0, essi combinano differenti tipi di dati da diverse sorgenti pubbliche, le informazioni raccolte vengono visualizzate all'interno del browser Web.

L'architettura dei sistemi di Mashup può essere di due tipi: *Web-Based* oppure *Server-Based*. La prima generalmente utilizza il browser Web per ricombinare e visualizzare i dati, mentre quella Server-Based analizza e combina i dati all'interno di un server Web inviando al browser i dati nella loro forma finale, pronti per essere visualizzati.
La parte della visualizzazione e della presentazione viene fatta all'interno del browser Web, le tecnologie utilizzate in questa parte sono HTML5[6], CSS[7], JavaScript ed AJAX[8]. Per le fonti di dati invece, troviamo le tecnologie e i formati previsti dalle diverse sorgenti: per i Web Service si accede ai dati tramite le API[9] pubbliche sfruttando le tecnologie SOAP[10], REST[11] e JSON[12], per i Feed si utilizzano i diversi standard previsti come RSS[13] o Atom.
La gestione, l'invio e la memorizzazione dei dati invece avviene utilizzando le tecnologie quali XML[14] o JSON.

[6]HyperText Markup Language
[7]Cascading Style Sheets
[8]Asynchronous Javascript And XML
[9]Application Programming Interface
[10]Simple Object Access Protocol
[11]REpresentational State Transfer
[12]JavaScript Object Notation
[13]Really simple syndication
[14]Extensible Markup Language

2.3 Tool che permettono di personalizzare il WEB

In questa sezione verranno presentati i tool analizzati durante la fase di studio della tesi.

2.3.1 Greasemonkey

Greasemonkey [1] è un'estensione del browser Mozilla Firefox che consente di utilizzare degli script che modificano le pagine web quando vengono caricate. Normalmente gli script sono scritti dallo sviluppatore del sito internet, mentre Greasemonkey permette all'utente di eseguirne alcuni e a propria scelta. Può essere utilizzato per aggiungere nuovi contenuti, migliorarne l'aspetto, semplificare operazioni, combinare dati con altre pagine e numerosi altri scopi. Gli utenti creano gli script, tipicamente utilizzando JavaScript per modificare il DOM della pagina.
In genere gli script hanno il formato nome.user.js ed all'interno del file sono contenute anche alcune meta-informazioni, che specificano l'autore, la versione e gli indirizzi dei siti per cui è pensato.
Tipicamente si utilizza questa estensione per:

- Aggiungere pulsanti o link che consentono di sfruttuare delle funzionalità esterne al sito oppure non facilmente raggiungibili tramite i comandi originali dell'applicazione (ad esempio aggiungendo tasti come sposta o cancella a WebMail);
- Comparazione di dati provenienti da siti diversi (ad esempio prezzi di negozi online);
- Rimuovere paragrafi o immagini (come pubblicità o spam);
- Spostare o modificare contenuti secondo l'esigenza dell'utente;
- Compilazione automatica di dati (come e-mail e nickname).

Scrivere uno script per Greasemonkey richiede pero' la conoscenza di HTML e JavaScript, escludendo così l'utente ordinario dal poterne scrivere una. Inoltre l'uso di Greasemonkey potrebbe portare alcuni problemi, ecco alcuni esempi:

1. Sostituire la pubblicità o il contatore di visite di un sito con il corrispettivo di un altro potrebbe falsare le statistiche o diminuire gli introiti per aumentare quelli del rivale;

2. Per comparare i dati di siti diversi è necessario accedere alle pagina web di tali siti, anche non si stanno effettivamente visitando, ma di cui ci interessano le informazioni, questo porta ad incrementare il traffico generandone uno fittizio;

3. Il cambiamento della forma di un sito potrebbe rendere il relativo script inutile o anche dannoso, creando anche degli errori, diventerebbe così necessario disattivare lo script e riscriverlo.

2.3.2 Chickenfoot

Chikenfoot [14] è un estensione di Mozilla Firefox che consente di customizzare le pagine Web senza che l'utente veda il codice HTML che sta all'interno della pagina utilizzando uno strumento, il browser, con cui l'utente ha familiarità. Chickenfoot si basa su tre principi:

1. viene eseguito direttamente dentro il browser, l'utente si rende conto in tempo reale di cosa sta editando in quanto la schermata contiene sia l'ambiente di sviluppo che la schermata risultato;

2. descrive una sintassi informale basata su parole che possono essere conosciute dall'utente come "*click*", "*enter*";

3. consente all'utente di descrivere le componenti della pagina in termini di che effetto producono nella pagina, ad esempio "*click* <*nome_bottone* >" identifica la pressione del bottone di etichetta <*nome_bottone*>.

E' necessario dunque conoscere un mini linguaggio molto semplice per poter creare gli script, in questo linguaggio è possibile utilizzare anche alcune primitive di JavaScript che gli sviluppatori hanno scelto di rendere fruibili a chi usa Chickenfoot.
I comandi principali ad alto livello sono:

- *click ("label")*: effettua il click di bottoni e hyperlink che hanno l'etichetta passata come parametro;
- *enter("campo", "valore")*: inserisce il valore nella textbox di etichetta "campo";
- *pick("elemento","valore")*: selezionare il valore di una listbox o dropdown list di etichetta "elemento";
- *keypress("campo", "valore")*: esegue la simulazione della tastiera per la textbox di etichetta "campo", inserendo i caratteri indicati dal parametro "valore", oppure simulando la pressione di combinazioni di tasti indicata dal parametro "valore";
- *go("address")*: accede all'indirizzo indicato dal parametro "address";

Questa forma di customizzazione consente l'automatizzazione delle azioni dell'utente, la registrazione di tale sequenza di azioni, ed infine la possibilità di utilizzare anche siti esterni a quello visitato e quindi creare dei mashups di informazioni.

2.3.3 CoScripter

CoScripter [6] è un sistema che permette all'utente di registrare, condividere e automatizzare dei task da eseguire nel Web. La presenza di un repository dove condividere gli script creati, rende possibile agli utenti di avere una base di partenza da dove fare esperienza nell'utilizzo e la programmazione degli script. Anche questa tecnologia si presenta come un'estensione per il browser Firefox ed è stata sviluppata dall'IBM. L'utente dopo aver installato

l'estensione su Firefox, può navigare sul repository per cercare script interessanti da cui partire. Il sito supporta la ricerca e l'ordinamento secondo vari parametri.
Una volta trovato lo script lo si installa all'interno della sidebar di CoScripter, dove troviamo un'interfaccia che consente di seguire l'esecuzione dello script linea per linea, ad ogni passo il sistema evidenzia il componente destinatario dell'azione che viene eseguita. Lo script può essere eseguito anche automaticamente.
Il sistema da la possibilità all'utente di registrare i propri script, innanzitutto si crea un nuovo script vuoto, poi si avvia la registrazione delle azioni cosiì l'utente crea lo script semplicemente eseguendo le azioni che intende automatizzare all'interno del browser. Lo script risultato viene salvato automaticamente all'interno del repository.
Il linguaggio di script ha la caratteristica di essere comprensibile all'utente e contemporaneamente interpretabile dal computer, perciò troviamo uno strumento che registra e capisce lo script cosi da riprodurre le azioni eseguite dall'utente. La registrazione funziona aggiungendo degli event listener per ogni possibile evento che si può eseguire sul DOM HTML.
Ad esempio un azione di click deve sempre avere la forma *"<click the TargelLabel TargetType>"* dove *"TargetType"* viene derivato dal tipo di nodo del DOM dove l'event listener ha registrato un azione, mentre *"TargetLabel"* viene estrapolato dal codice HTML usando delle euristiche.
L'attuale versione di CoScripter utilizza un approccio basato su una grammatica per il parsing dei passi dello script. I passi dello script vengono parsati utilizzando un parser LR(1) che trasforma la rappresentazione testuale del passo in un oggetto che rappresenta un comando Web, comprendente i parametri necessari per interpretare tale comando in una data pagina Web (come ad esempio la labe le il type dell'elemento target).
Il linguaggio di scripting di CoScripter ha bisogno di algoritmi specializzati per registrare le azioni dell'utente e riprodurle, tali algoritmi si basano su un insieme di euristiche, sia per generare una label comprensibile all'utente per il

relativo componente interattivo nella pagina, sia per trovare il relativo componente data una determinata label. CoScripter mette a disposizione una grande quantità di euristiche che analizzano il DOM e rendono disponibili le label desiderate per una grande quantità di elementi.

2.3.4 Clip, Connect, Clone

Clip, Connect, Clone [10] è il prototipo di uno strumento che consente all'utente di creare interfacce personalizzate basandosi su 3 linee guida:

1. *Clip*: l'utente seleziona e taglia gli elementi di input/output che gli interessano in una determinata pagina Web e li importa nelle celle di un foglio di calcolo in stile Excel chiamato C3Sheet;

2. *Connect*: l'utente collega il contenuto delle celle usando delle formule, questo abilita il collegamento tra più applicazioni;

3. *Clone*: si possono clonare le celle per fare in modo che gli elementi contenuti all'interno di esse possano essere utilizzati contemporaneamente con valori distinti.

La fase di clipping viene realizzata tramite un estensione per il browser (in questo caso gli sviluppatori hanno utilizzato Microsoft Internet Explorer), l'utente può selezionare gli elementi che gli interessano, anche regioni estese della pagina, ed effettuare il drag and drop verso il 3CSheet per definire una nuova cella.
La cella conterrà la porzione di pagina selezionata mantenendo intatte le dimensioni e lo stile della pagina originale. L'applicazione mantiene anche i riferimenti logici agli elementi selezionati, ad esempio se si selezionano alcuni elementi di una form, oppure degli elementi che vengono generati durante la navigazione, il sistema conosce la loro posizione e le loro dipendenze all'interno della pagina iniziale. In caso di aggiornamento dei valori collegati a quell'elemento il sistema genera dei valori consistenti per l'elemento stesso.
La fase di connessione è simile alla gestione dei fogli di calcolo, si definiscono

delle formule per realizzare delle procedure, in maniera tale da sfruttare le relazioni tra le celle per controllare le connessioni tra le parti di pagine Web contenute all'interno delle celle stesse.
La fase di cloning consente di generare diversi scenari all'interno dell'applicazione, in particolare è possibile condividere parti del contenuto Web precedentemente selezionato, come gli elementi di input, e duplicare l'elemento (contenuto in una cella) che vogliamo condividere.
Questa funzione ci permette di poter utilizzare lo stesso contenuto Web in due procedure diverse con valori differenti. Lo strumento non genera veri e propri mashup, infatti non crea nuove applicazioni Web, ma delle applicazioni che possono essere usate in locale.

2.3.5 Intel Mash Maker

Intel Mash Maker [13] è uno strumento che consente all'utente di personalizzare e migliorare una pagina Web realizzando quello che viene detto "*overlay mashup*", cioè l'informazione creata viene aggiunta in determinati punti della pagina web originale.
Anche questo strumento nasce come un estensione del browser Firefox ed Internet Explorer, l'interazione avviene attraverso una toolbar nel browser. Mash Maker usa un'architettura a tre livelli per creare i mashup:

- *Wrappers*: estraggono l'informazione strutturata dalle pagine Web;
- *Widgets*: interrogano le informazioni estratte dalle pagine e pubblicano nuove informazioni e visualizzazioni;
- *Mashups*: specificano quali widget devono essere aggiunti ad una pagina, con i rispettivi settings ed l'aspetto con il quale devono essere integrati in una determinata pagina.

Il tipo di mashups che viene realizzato è di tipo "*overlay mashup*" per le seguenti ragioni:

1. Rendere chiaro che operazioni esegue il mashup: l'utente che utilizza e applica il mashup alla pagina che sta visitando non conosce come agisce il mashup. Siccome la differenza tra la pagina con o senza il mashup è immediatamente visibile all'occhio dell'utente, egli può facilmente vedere l'effetto e decidere se gli è utile o meno;

2. Prevenire cattivi comportamenti del mashup: il fatto che l'utente esegua dei mashups senza sapere cosa essi fanno, fa si che sia necessario limitare le occasioni in cui possono accadere dei comportamenti inattesi e dannosi per la navigazione dell'utente;

3. Ridurre i problemi legali: ci sono diverse problematiche legate alla proprietà dei contenuti di un sito, ad esempio "se un mashup può modificare una pagina arbitrariamente, cosa succede se il proprietario della pagina non approva tali modifiche?"

   L'overlay mashup non modifica il contenuto originale della pagina ma aggiunge del contenuto che si distingue visivamente dalla pagina originale.

2.3.6 iGoogle

iGoogle [2] (nato *Google Personalized Homepage*), è un servizio di Google. Consiste in una pagina Web che da la possibilità all'utente di avere a disposizione delle informazioni di natura e fonte diversa visualizzate all'interno di widget chiamati "*iGoogle gadget*"; tali informazioni spaziano dalle notizie, alle foto, fino ai video, ma è possibile avere anche informazioni personali come la vista della propria casella di posta, del proprio indirizzo Ip, etc.
L'utente ha cosi la possibilità di crearsi una propria pagina con le informazioni che desidera eventualmente, impostandola come home page da visualizzare all'avvio del browser. La pagina suddivide i widget in 3 colonne e l'utente è libero di sistemarli all'interno di esse a suo piacimento.

2.3.7 Yahoo! Pipes

Yahoo! Pipes [5] è un'applicazione Web creata da Yahoo! che consente all'utente di costruire dei mashup di dati provenienti da fonti diverse (tipicamente Feed RSS, news o informazioni provenienti da servizi web) per creare delle nuove applicazioni che utilizzano tali dati ed infine, pubblicare tali applicazioni e renderle disponibili sulla rete.
L'editor web consente all'utente di costruire le proprie applicazioni utilizzando un editor cosi composto: in centro si trova il Canvas dove dovrà sistemare i vari moduli che costituiranno la pipe; sulla sinistra dell'editor si trova la Library dei componenti (chiamati modules) che possono essere draggati all'interno del canvas. I componenti utilizzabili sono:

- *Sources*: troviamo le componenti che si usano per prelevare i dati da una o più sorgenti nel web;
- *User Inputs*: sono componenti che consentono l'input dell'utente durante l'esecuzione della pipe, l'utente può inserire diversi tipi di input: date, luoghi, numeri, stringhe oppure URL;
- *Operators*: i componenti di questa categoria consentono di filtrare o trasformare i dati che attraversano la pipe, le operazioni disponibili sono: *filter, count, location extractor, loop, regex, rename, reverse, sort, split, sub-element, tail, truncate, union, unique* e *web service*;
- *URL*: questo modulo consente di corstruire delle URL automaticamente generandole in base ai parametri inseriti in questo modulo;
- *String*: sono moduli che si occupano della manipolazione delle stringhe: string builder, espressioni regolari, suddivisioni, estrazione di termini, traduzioni, ecc;
- *Date*: ci sono due moduli:

  - date builder: converte un testo in una data (es. passando in input la stringa "*yesterday*" viene restituito un oggetto con la data di ieri);
  - date formatter: prende una data come input e la restituisce nel formato desiderato.

- *Location*: il modulo di questa categoria converte una stringa passata come input nel corrispondente luogo geografico;
- *Number*: il modulo di questa sezione compie solo semplici operazioni matematiche sul numero in input e restituisce il risultato.

Infine sul lato inferiore c'è il Debugger: quando l'utente clicca su un modulo si vedere in tempo reale il risultato dell'esecuzione del modulo selezionato.

2.3.8 WSO2 Mashup Server

The WSO2 Mashup Server [4] è una piattaforma di mashup open source che supporta mashups scritti in JavaScript. L'utente scrive i propri script che vengono mandati in esecuzione sul server, una volta effettuato il deploy di tali script, un utente utilizzatore può usufruire delle funzionalità degli script invocandoli come un Web Service.
L'autore degli script può comporre all'interno di essi dati e informazioni provenienti da diverse fonti in maniera simile a quanto avviene nelle Yahoo Pipes, combinarli e modificarli a suo piacimento per creare dei mashups. L'utente utilizzatore invece non si deve preoccupare di come l'output viene costruito, semplicemente invoca lo script inserendo eventuali parametri di input.
A differenza di Yahoo Pipes lo sviluppatore non ha a disposizione nessun editor grafico, ma deve sviluppare lo script con l'editor testuale.

2.3.9 Microsoft Popfly

Microsoft Popfly [3] è un sistema per la creazione di applicazioni Web, giochi, mashup che si sfruttava il runtime di Microsoft SilverLight. Il progetto è stato dismesso nell'agosto del 2009.
I moduli presenti in questa applicazioni erano i seguenti:

- *Game Creator*: permetteva la creazione di giochi che potevano essere condivisi su Facebook oppure usati come Windows Live Gadget.

- *Web Creator*: consentiva la creazione di pagine Web senza dover scrivere direttamente il codice HTML, e integrare facilmente i mashup prodotti con la piattaforma.

- *Mashup Creator*: consentiva la creazione di mashup, l'utente poteva connettere dei blocchi predefiniti visualizzati graficamente come dei cubi, ogni blocco aveva una serie di parametri di input che si potevano configurare. Una volta selezionati i componenti da connettere, l'utente poteva creare delle connessioni tra i blocchi collegandoli graficamente tra di loro, ed indicando quali parametri voleva mettere in relazione tra i vari blocchi.
  Funzionalità avanzate consentivano agli utenti più esperti di aggiungere del codice JavaScript o HTML per modificare il comportamento dei blocchi e affinare il comportamento dell'applicazione creata.

Era presente un social network interno che si chiamava *Popfly Space* dove l'utente poteva salvare, condividere e votare le applicazioni costruite. La piattaforma permetteva inoltre all'utente di scaricare i mashup creati all'interno della Microsoft Sidebar di Windows, oppure di intergrarli all'interno di Windows Live Spaces. Un ulteriore feature consisteva nell'integrazione con Microsoft Visual Studio Express ciò consentiva all'utente di modificare i propri mashup all'interno dell'ambiente di sviluppo.
Nel luglio 2009 la Microsoft ha deciso di cancellare il progetto e nell'agosto seguente tutte le risorse riguardanti il progetto furono rimosse dalla rete.

2.3.10 EzWeb Enterprise Mashup

EzWeb Enterprise Mashup [9] è una piattaforma per il mashup sviluppata da Telefonica, essa consente di utilizzare una serie di *gadget* che l'utente può connettere tra di loro per creare e modellare le proprie applicazioni, il target è quello di creare una piattaforma che possa essere utilizzata all'interno delle aziende.
La piattaforma mette a disposizione dell'utente un'insieme di gadget predefiniti all'interno di un *catalogue*. La connessione tra i gadget da parte dell'utente porta il nome di *piping composition* che crea un flusso di esecuzione tra i gadget collegati, la piattaforma dal suo lato utilizza la *wiring composition* che consente di avere in esecuzione più pipe contemporaneamente tra i gadget selezionati.

2.3.11 d.mix

Il tool *d.mix* [7] consente di creare delle applicazioni basate sui Web service, attraverso un *site-to-service map*.
Attraverso d.mix l'utente naviga in siti Web che sono stati annotati, e seleziona gli elementi della pagina ai quali è interessato. La piattaforma genererà così il codice per ottenere tali elementi attraverso chiamate ai Web service messi a disposizione dal sito visitato.
Tale codice può essere modificato, eseguito e condiviso attraverso un wiki. La piattaforma mette a disposizione una serie di esempi che possono essere modificati e composti dall'utente per creare le proprie applicazioni. L'utente finale deve però avere nozioni di programmazione per poter modificare tale codice.

2.3.12 Crowdsourcing Platform and CrowdDesign

Si tratta di un prototipo [12] che permette uno sviluppo leggero di sistemi Web, basato sull'utilizzo di *crowdsourcing* per la fase di progettazione e il processo di sviluppo.

La piattaforma è stata pensata per favorire lo sviluppo di interfacce Web basate sui dati, combinandole con il paradigma *plug and play*. Gli utenti ai quali è destinata la piattaforma vanno dagli sviluppatori agli end users.
Le applicazioni vengono create combinando delle componenti provenienti da diversi siti Web, tali componenti vengono sviluppate ed acquisiscono funzionalità attraverso il contributo degli utenti. I componenti infatti non devono essere pre-esistenti o progettati solo da chi sviluppa l'applicazione, ma possono anche essere progettati on-demand richiedendo la partecipazione degli altri utenti, che contribuiscono fornendo più soluzioni alternative.
L'ambiente mostra allo sviluppatore dell'applicazione le diverse soluzioni che sono state pensate dagli altri utenti per quel componente, dando la possibilità di scegliere la soluzione che egli preferisce.

2.3.13 Confronto tra gli strumenti analizzati

In questa sezione viene effettuato un confronto tra gli strumenti analizzati. All'interno della tabella 2.1 sono elencate le proprietà possedute dai vari tool, in ogni riga sono presenti le informazioni relative ad uno strumento, mentre le colonne contengono le seguenti informazioni:

- *Tool*: nome dello strumento;
- *Tipologia utente*: tipologia dell'utente al quale fa riferimento lo strumento;
- *Mashup*: lo strumento consente la costruzione di mashup;
- *Share*: lo strumento consente la condivisione delle applicazioni create;
- *Collaboration*: lo strumento consente di sviluppare le applicazioni tramite la collaborazione degli utenti.

Tabella 2.1: Confronto tra gli strumenti

Tool	Tipologia utente	Mashup	Share	Collaboration
Greasemonkey	Script programmers	Si	Si	No
Chickenfoot	Script programmers	Si	Si	No
CoScripter	All Users	Possibili	Si	Si
Clip, Connect, Clone	All Users	Possibili	Si	No
Intel Mash Maker	All Users	Si	Si	No
iGoogle	All Users	Si	Si	No
Yahoo! Pipes	All Users	Si	Si	No
WSO2 Mashup Server	Script programmers	Si	Si	No
Microsoft Popfly	All users	Si	Si	No
EzWeb	All Users	Si	Si	Si
d.mix	All Users	Si	Si	Si
CrowdDesign	All Users	Si	Si	Si

La prima grande distinzione tra i tool analizzati è dovuta alla tipologia degli utenti che posso creare delle applicazioni tramite il loro utilizzo, *GreaseMonkey*, *Chickenfoot* e *WSO2 Mashup Server* infatti richiedono la conoscenza di un linguaggio di scripting (JavaScript), mentre gli altri tool possono essere utilizzati da tutti i tipi di utenti senza conoscere particolari linguaggi di programmazione.
Tutti i tool consentono agli utenti di condividere le proprie applicazioni, infatti ogni tool possiede un repository di supporto all'interno dei quale gli utenti possono inserire le proprie applicazioni per metterle a disposizione degli altri utenti.
Per quanto riguarda la creazione di mashup tutti i tool ne consentono la composizione seppure con qualche limite, infatti *CoScripter* permette la creazione di applicazioni che importano il contenuto di siti esterni all'interno del sito dove agisce lo script, mentre *Clip, Connect, Clone* pur consentendo di utilizzare diverse sorgenti di dati genera come output un applicazione desktop,

per cui non si tratta di veri e propri mashup.
L'ultimo aspetto considerato riguarda la presenza di strumenti per favorire lo sviluppo di applicazioni tramite la collaborazione tra gli utenti. *CoScripter* consente di modificare liberamente gli script degli altri utenti, infatti quando l'utente esegue uno script prelevato dal wiki condiviso i comandi che lo compongono vengono visualizzati all'interno dell'estensione nel browser.
L'utente può modificare lo script agendo direttamente all'interno dell'estensione, modificando le istruzioni che lo compongono. *EzWeb*, *d.mix*, *CrowdDesign* supportano la collaborazione tra gli utenti che divengono quindi una componente importante del processo di sviluppo di un mashup.
Un utente può richiedere la collaborazione degli altri utenti per sviluppare le varie parti della sua applicazione, per revisionarla e valutarla.

Capitolo 3

Background

In questo capitolo verrà presentato il Mashup Editor, un ambiente grafico per EUD di mashup di applicazioni Web, la caratteristica principale dell'editor è quella di consentire agli utenti la manipolazione diretta delle parti di applicazioni Web che intendono utilizzare per comporre i propri mashup.

3.1 Architettura della piattaforma

La struttura dell'ambiente è semplice si tratta di una parte client-side l'*EUD Environment*, e di una parte server-side il *Proxy/Mashup Server*. Quest'ultimo funziona come Proxy quando si effettua l'accesso alle applicazioni che si desidera includere all'interno del mashup, infatti si occupa di inserire alcuni script Javascript che abilitano la selezione delle parti di applicazione che faranno parte del nuovo mashup. Funziona invece come Mashup Server quando fornisce il supporto per l'ambiente grafico utilizzato dall'utente. L'EUD environment consente di creare le connessioni tra le componenti che originariamente facevano parte di applicazioni diverse, e di gestire i propri mashup creati.
Nella piattaforma di mashup l'accesso alle applicazioni Web passa dunque attraverso il Proxy/Mashup server, che analizza il documento originale e aggiunge gli ID a quegli elementi rilevanti che non ne hanno uno. Gli elementi

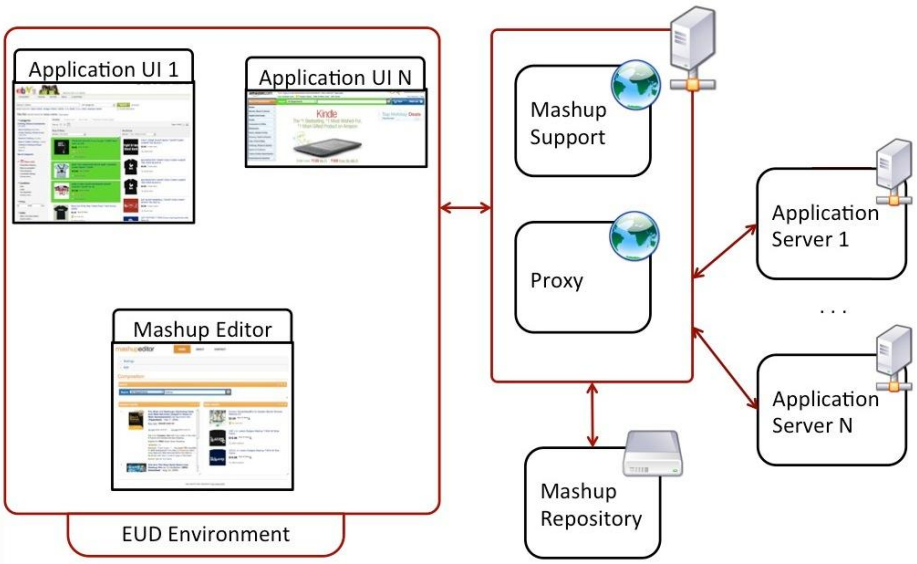

Figura 3.1: Struttura della piattaforma di mashup

rilevanti sono quelli che possono essere selezionati in modo interattivo (come DIV, FORM, INPUT, TABLE, ...).
Il server include una serie di script, che vengono sfruttati dal lato client, in tutte le pagine Web a cui si accede attraverso di esso. Questi script consentono agli utenti di selezionare in modo interattivo gli elementi da includere nel mashup appena creato. Se l'elemento è un contenitore (ad esempio un div) suoi componenti interni sono ricorsivamente compresi nella selezione. Gli elementi selezionati nella pagina Web sono evidenziati cambiando il colore di sfondo.
La parte client-side della piattaforma è stata implementata utilizzando le tecnologie HTML + JavaScript, in particolare si sono utilizzate le librerie *JQuery* e *JqueryUI*. Il lato server è stato realizzato utilizzando Java Servlet e JSP, il Webserver è Apache Tomcat e la piattaforma funziona su Windows, Unix e Mac OS.

3.2 Selezione delle componenti

La selezione delle componenti delle applicazioni di partenza sta alla base della creazione di un nuovo mashup, a tal fine gli script che vengono automaticamente inclusi dal proxy server permettono di gestire gli eventi *onMouseOver* e *onClick*.
Se sono già definiti degli *handler* per tali eventi nell'applicazione Web, vengono conservati e gli handler necessari alla selezione sono aggiunti a quelli già presenti. In particolare, al passaggio del mouse (evento *onMouseOver*) si evidenzia l'elemento sopra il quale si trova il mouse settando un colore di sfondo grigio, mentre i componenti selezionati sono evidenziati dal colore verde.
Un componente viene evidenziato cambiando il suo colore di sfondo e colore del bordo. Il valore originario di questi attributi viene salvato su una variabile JavaScript per poter essere ripristinato quando il componente è deselezionato oppure il mouse esce dal suo interno.
Quando l'utente è soddisfatto della sua selezione invia ciò che ha scelto al MashupEditor, parallelamente quindi il Mashup Support riceve dal proxy il DOM e la lista degli IDs degli elementi selezionati, il DOM inviato è *state-persistent* (es. vengono conservati i valori inseriti dall'utente all'interno di un form).
All'interno del server vengono eliminati dal DOM tutti quelli elementi che non sono stati selezionati dall'utente, mentre quelli selezionati vengono inviati all'editor che mostra all'utente le parti dell'applicazione che ha selezionato all'interno di un Mashup Widget.
Un Widget mostra all'utente esattamente quello che ha selezionato, conservando anche l'aspetto, le dimensioni e lo stato che gli elementi avevano all'interno dell'applicazione originale. L'utente deve anche indicare un titolo per il widget che ha creato.
La piattaforma memorizza all'interno di un descrittore le informazioni che definiscono un widget, la sua posizione e dimensioni all'interno dell'editor, il path dei file HTML che sono stati creati all'interno del mashup server, l'elen-

co dei parametri di input che l'utente può inserire e che possono essere inviati ad altri componenti, inoltre si memorizza l'elenco degli ID degli elementi che l'utente ha selezionato, così è possibile per l'utente aggiornare il contenuto del widget, come verrà spiegato in seguito.
La sequenza delle azioni che avvengo all'interno del MashupEditor sono schematizzate nella seguente figura:

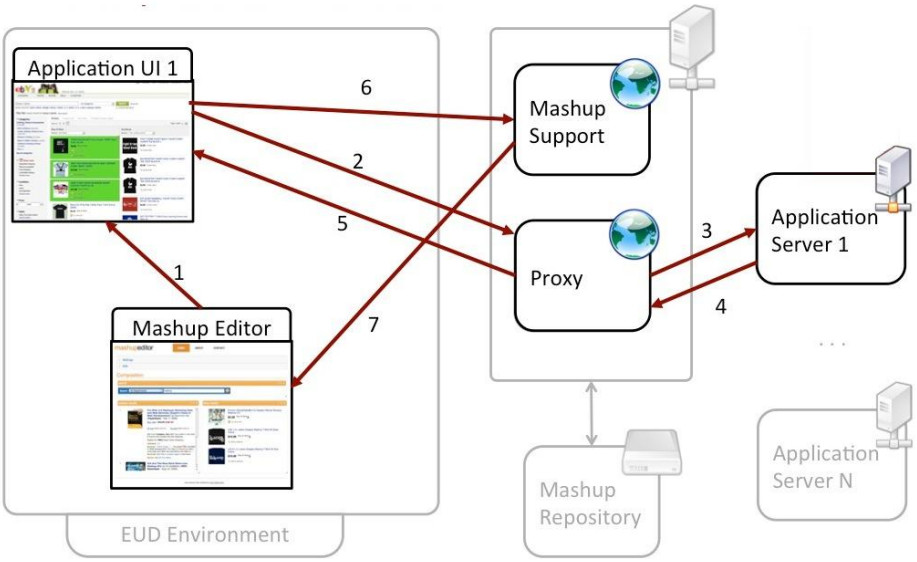

Figura 3.2: Azioni che vengono eseguite per la creazione di un widget

1. L'utente inserisce la URL che vuole visitare all'interno dell'interfaccia utente del MashupEditor. Una nuova finestra viene aperta nel browser Web.

2. La nuova finestra invia una richiesta HTTP al proxy.

3. Il proxy si collega con il server dove risiede l'applicazione scelta.

4. Il proxy riceve la risposta dal server dell'applicazione ed aggiunge gli script per il supporto alla selezione diretta dei componenti nell'interfaccia utente.

5. Il proxy invia la risposta alla nuova finestra del browser Web.

6. L'utente seleziona un insieme di elementi dall'interfaccia utente e invia la selezione alla piattaforma per creare un nuovo widget. Il proxy invia il DOM e la lista degli IDs degli elementi selezionati al Mashup Support che estrae i componenti selezionati.

7. Il Mashup Support comunica all'interfaccia utente del MashupEditor che l'utente ha creato un nuovo widget. L'utente inserirà quindi il nome del widget appena creato all'interno di una finestra di dialogo, l'applicazione mostrerà le parti dell'applicazione selezionate all'interno di un widget dedicato.

La seguenti figure mostrano le azioni compiute dall'utente per creare un widget.

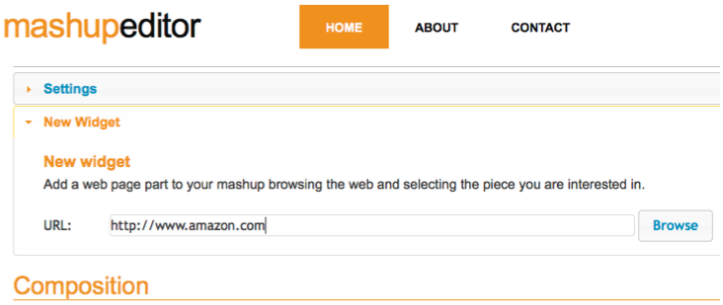

Figura 3.3: Navigazione verso il sito Web

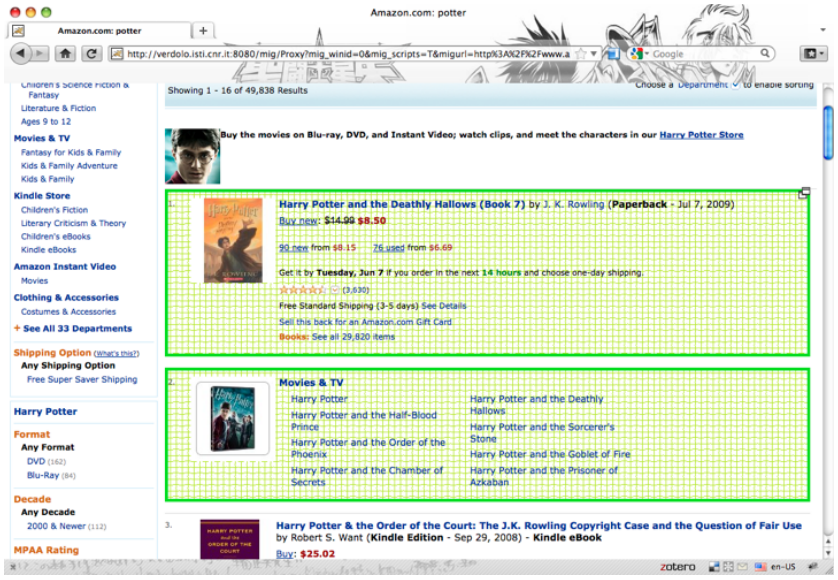

Figura 3.4: Selezione delle parti di applicazione

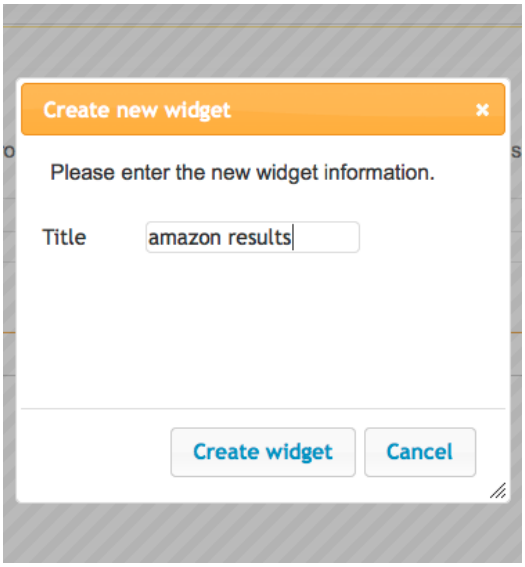

Figura 3.5: Inserimento del titolo del nuovo widget

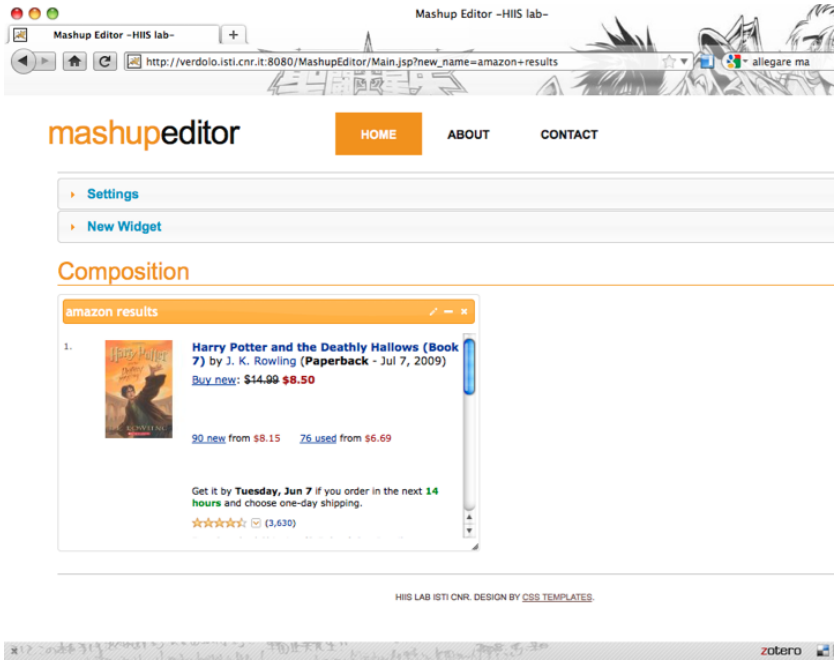

Figura 3.6: Visualizzazione del nuovo widget all'interno del MashupEditor

3.3 Riconoscimento dei parametri di input

Per creare delle comunicazioni tra i widget, la piattaforma permette di effettuare il riconoscimento degli elementi che controllano le comunicazioni dei widget stessi, i quali si possono dividere in due gruppi:

- Widget che richiedono un input da parte dell'utente e che possono produrre un output che viene inviato ad un Web server detti *input components*;
- Widget che richiedono un input da parte del Web server e che utilizzano tale input per produrre dell'output da mostrare all'utente detti invece *output components*.

La piattaforma è in grado di generare delle connessioni tra questi tipi di componenti, anche se inizialmente appartenevano ad applicazioni diverse. L'utente può visualizzare l'elenco dei parametri che determinano il contenuto di ogni widget di tipo *output component* e selezionare gli elementi di input che egli ha utilizzato per ottenere quel determinato widget.
Gli altri parametri visualizzati sono quelli che le applicazioni generano per controllare lo stato dell'applicazione stessa.

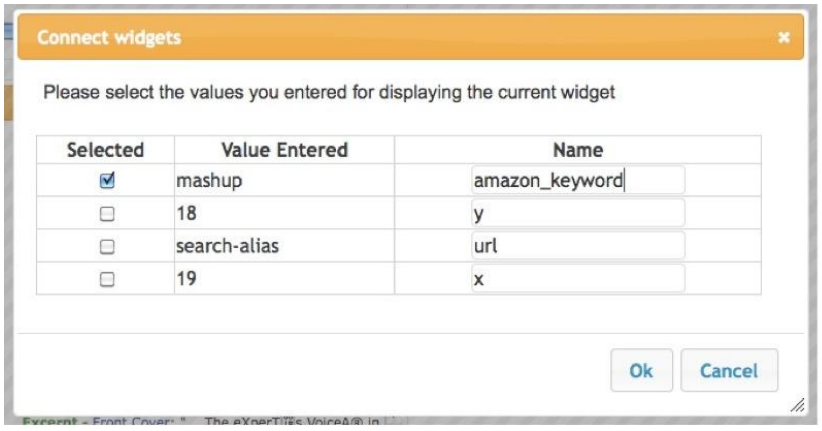

Figura 3.7: Schermata di riconoscimento dei parametri

Questo è possibile perché la piattaforma analizza il metodo con cui viene effettuata la richiesta HTTP durante la navigazione tramite il proxy, si procede quindi ad analizzare il contenuto della query string se è stata effettuata una chiamata GET, o il contenuto della post in caso di chiamata POST.
Poichè i parametri sono codificati secondo una sintassi standard (URL encoding), è possibile sfruttare l'analisi automatica di essi e mostrarli all'utente per il riconoscimento.

3.4 Connessione tra i widget

Per connettere tra loro i vari widget, l'utente attiva il gestore delle connessioni, il quale consente all'utente di indicare quali parametri degli *input components* devono essere utilizzati per fornire i parametri per determinare il contenuto degli *output components*.
All'interno di una finestra di dialogo viene mostrato all'utente l'elenco degli *output components* disponibili e quali sono gli elementi di input che li controllano, l'utente può in questo modo scegliere di associarli e indicare alla piattaforma che il valore inserito in un determinato *input component* dovrà essere utilizzato per riempire il parametro di input selezionato. La finestra di dialogo mostra all'utente l'elenco dei campi di input che sono contenuti in un widget (es. text fields, drop down lists, text areas etc.).
Per ognuno di essi la piattaforma mostra l'elenco degli *output components* che possono ricevere il valore a l'utente può inserire tramite quell'elemento di input.

Nella figura 3.8 si può vedere un esempio della procedura di connessione (nella parte superiore della figura troviamo il widget da connettere, mentre nella parte inferiore la finestra di dialogo da dove effettuare la connessione):

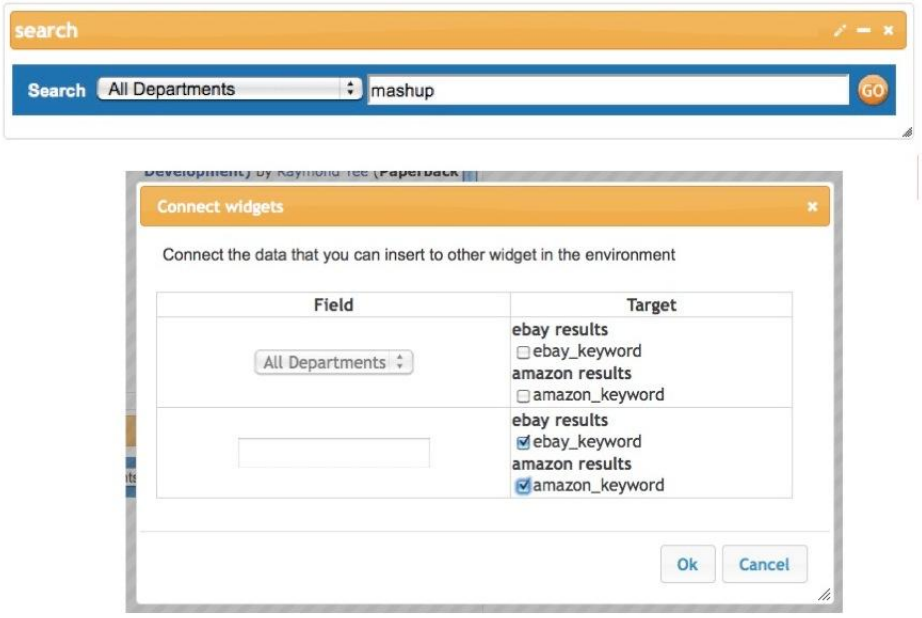

Figura 3.8: Schermata della finestra per la connessione tra widget

Gli elementi di input sono due, una drop down list e un text field. La drop down list non è connessa ad alcun parametro, mentre il valore inserito nel text field viene connesso al parametro *amazon_keyword* del widget *Amazon Result* ed al parametro *ebay_keyword* del widget *Ebay Result*. Questo significa che quando verrà modificato il valore del text field, e fatta la submit della form l'editor aggiornerà il contenuto dei widget collegati. Il valore inserito dall'utente nel textfield sostituito all'interno dei parametri corrispondenti ad *amazon_keyword* e *ebay_keyword* all'interno della query string oppure del post content.

3.5 Aggiornamento del mashup

La procedura di aggiornamento di un mashup viene avviata dall'utente inserendo dei nuovi valori all'interno di un widget di input e inviandoli alla piattaforma. La procedura di aggiornamento segue i seguenti passi:

- La piattaforma raccoglie i valori inseriti dall'utente all'interno del form;
- Per ogni elemento di input analizza quali sono i widget di output che devono ricevere tale input e li segnala come da aggiornare;
- Viene aggiornato il valore del parametro individuato all'interno della query string o del post content;
- Per ogni componente da aggiornare viene effettua la chiamata all'application server scaricando l'intera pagina Web, filtrando poi il DOM utilizzando la lista degli IDs degli elementi che l'utente aveva selezionato durante la navigazione, quindi aggiorna il contenuto all'interno del mashup editor.

La seguenti figure mostrano il risultato dell'operazione di aggiornamento del mashup, nella figura 3.9 vediamo il mashup creato cercando la parola chiave "*mashup*", mentre nella figura 3.10 viene mostrata la schermata dell'editor dopo aver ricercato la parola chiave"*Potter*" all'interno della casella di ricerca unica.

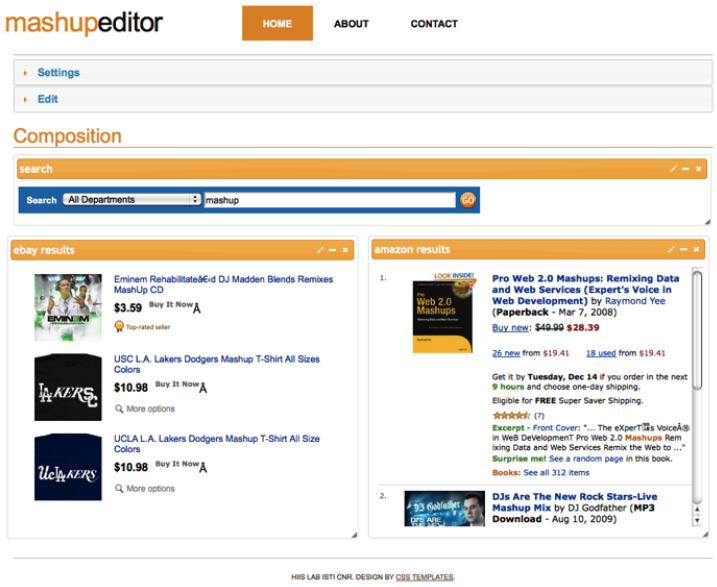

Figura 3.9: Risultato finale della creazione di un mashup

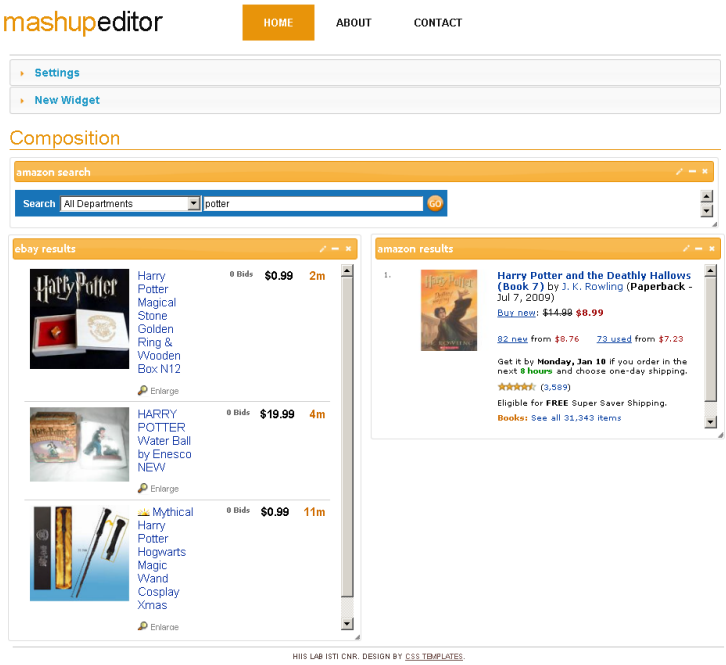

Figura 3.10: Aggiornamento del mashup

3.6 Limitazioni della versione precedente

Analizzando le funzionalità della piattaforma di mashup sono stati individuati i seguenti punti che sarebbero utili da affrontare per migliorarla:

- modificare il sistema di creazione di un widget tramite la navigazione Web. Ora l'utente per creare un widget deve inviare alla piattaforma di mashup sia l'*input component* che l'*output component* separatamente, sarebbe invece utile un meccanismo che permetta all'utente di creare il widget con un unico comando;

- il riconoscimento dei parametri di un widget può risultare complesso per gli utenti che non hanno familiarità con il concetto di parametro, sarebbe meglio che la piattaforma riconosca gli input inseriti dall'utente automaticamente durante la navigazione Web;

- la fase di connessione dovrebbe essere modificata evitando, di far selezionare all'utente tramite la finestra di connessione i campi di input che vuole connettere con i parametri degli *output components*, studiando un sistema *user-friendly* che registri le azioni che l'utente vorrebbe compiere per connettere due widget.
  Ad esempio, se un valore deve essere copiato da un text field ad un altro, la definizione della connessione viene fatta dall'utente scrivendo un valore nel primo text field, e poi facendo un copia/incolla sul secondo campo;

- aggiungere la gestione degli utenti e delle loro aree di lavoro, dando ad essi la possibilità di salvare e poi ricaricare i loro progetti di mashup;

- creare un sistema per la condivisione e valutazione dei mashup tra gli utenti della piattaforma;

- integrare il MashupEditor con il social network Facebook, portando all'interno della piattaforma le funzionalità più utilizzate dagli utenti del social network.

Capitolo 4

Ambiente EUD per mashup di applicazioni Web

Nei prossimi tre capitoli verrà presentato il lavoro che ho svolto durante la tesi, strutturato come segue:

- *fase 1*: Progettazione e sviluppo delle funzionalità per trasformare l'ambiente grafico della piattaforma in un EUD environment;
- *fase 2*: Inserire aspetti collaborativi tra gli utenti della piattaforma di mashup, implementando le funzionalità per la condivisione e la collaborazione nello sviluppo dei mashup;
- *fase 3*: Integrazione del MashupEditor con il social network Facebook.

4.1 Introduzione

Dall'analisi di diversi documenti e dei risultati dei test utente, effettuati sulla versione precedente della piattaforma, sono state individuate una serie di modifiche necessarie per potenziare le funzionalità di EUD del MashupEditor:

- Migliorare il sistema di creazione dei widget tramite la navigazione Web;

- Modificare il sistema delle connessioni;
- Inserire un sistema per visualizzare le connessioni effettuate;
- Aggiungere la gestione degli utenti;
- Implementare il salvataggio e il caricamento dei mashup degli utenti;
- Aggiungere una libreria per il salvataggio dei widget creati.

4.2 Modifiche alla creazione dei widget

Come illustrato nel capitolo precedente la distinzione principale dei widget in due categorie (*input components* e *output components*) implicava che l'utente per utilizzare le parti di una stessa applicazione doveva costruire due widget, uno contenente la form di ricerca (*input component*) ed un altro che conteneva i risultati della ricerca (*output component*) e ricordandosi di inviare entrambe le selezioni al MashupEditor.
Inoltre l'utente doveva essere in grado di ricordare ed individuare quali erano i valori che aveva inserito per effettuare la ricerca, identificando cosi il corrispondente parametro che era stato settato, ed indicarli alla piattaforma che, durante la fase di creazione delle connessioni, li utilizzava per proporre all'utente i possibili collegamenti tra gli *input components* e gli *output components*.
Per prima cosa mi sono occupato di modificare la procedura di creazione dei widget, rendendo automatica e trasparente all'utente la creazione dell'*input component*. Ora vengono generati dei widget che comprendono sia l'*input component* che l'*output component*, questa modifica ha eliminato la necessità di far individuare e ricordare all'utente i valori dei parametri che aveva inserito.
Quando l'utente vuole creare un nuovo widget, durante la navigazione tramite il proxy server, dovrà selezionare i campi delle form di ricerca che vuole utilizzare (text field, check box, buttons, ecc...), inserire l'input, quando poi farà

la submit, la selezione verrà automaticamente inviata al Mashup Support che crea e conserva la parte del widget corrispondente all'*input component*. L'utente può continuare la sua navigazione selezionando i risultati della ricerca effettuata ed inviarli alla piattaforma. Il Mashup Support salva automaticamente l'elenco degli elementi di input compilati dall'utente durante la fase di ricerca, e crea una connessione interna al widget tra l'*input component* e l'*output component*.
Nelle figure 4.1 e 4.2 vediamo le differenze nell'aspetto del widget di Amazon tra la vecchia versione e la nuova versione.

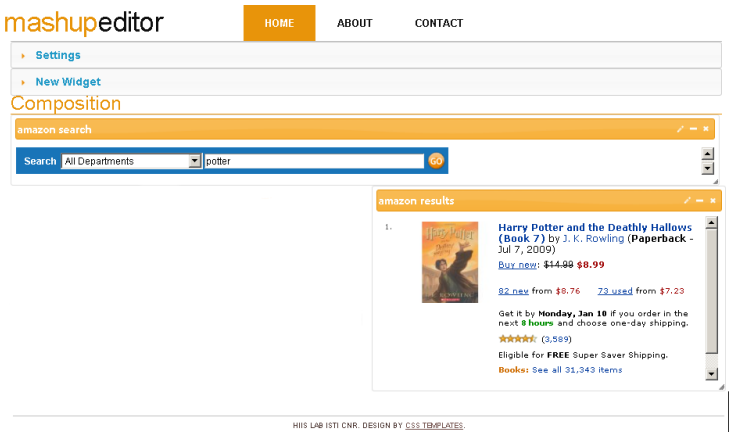

Figura 4.1: Aspetto del widget nella vecchia versione

Figura 4.2: Aspetto del widget nella nuova versione

Ogni widget ora è la rappresentazione dell'applicazione originale all'interno dell'ambiente grafico di sviluppo, il Mashup Support conserva infatti lo stato che l'applicazione originale aveva al momento della selezione da parte dell'utente. Inoltre, ogni widget può essere aggiornato ripetendo la ricerca effettuata, basta cambiare i valori degli input (es. inserire una nuova keyword all'interno dei text field) e il Mashup Support aggiornerà il contenuto.
L'utente non dovrà più occuparsi del riconoscimento dei parametri inseriti, questo consentirà anche agli utenti che non possiedono concetti di gestione dei parametri delle chiamate HTTP di poter creare i propri widget pronti per effettuare le connessioni.

4.3 Modifiche alla gestione delle connessioni

La modifica alla creazione dei widget ha portato alla progettazione di un nuovo sistema di connessioni tra i widget. Il nuovo sistema doveva essere efficiente come il precedente, ed allo stesso tempo usabile dagli *end user*.
La nuova struttura interna dei widget, che comprendono sia l'*input component* che l'*output component*, consente di lavorare all'interno dell'ambiente grafico su un insieme di widget indipendenti che riproducono esattamente il comportamento dell'applicazione Web da cui derivano.
Avendo a disposizione un *input component* per ogni widget, si è scelto di spostare la fase di connessione direttamente tra gli elementi di input che compongono tali *component*. Questo significa, dare la possibilità all'utente di collegare direttamente gli elementi di input che aveva selezionato durante la navigazione Web.
Il primo tipo di elementi di input sul quale è stato attivato questo nuovo meccanismo sono i "*text field*", per connettere due widget che dispongono di questi elementi di input, è stato scelto di utilizzare un paradigma conosciuto da tutti gli utenti dei Pc: il "*copia-incolla*".
L'utente che vuole collegare due widget Wid_i e Wid_j non deve far altro che copiare il contenuto di un text field del widget Wid_i, all'interno di un text

field del widget Wid_j.
Per supportare questa funzione è stato aggiunto alla piattaforma un sistema per registrare le azioni compiute durante la fase di connessione. Si utilizzano degli script, che definiscono degli *event handler* sugli elementi di input dei widget, per ascoltare gli eventi di tastiera e mouse dell'utente.
L'utente attiva la registrazione e la piattaforma memorizza tutte le azioni che compie sugli input dei widget tramite mouse e tastiera, una volta fermata la registrazione il Mashup Support genera automaticamente le connessioni tra i widget coinvolti.
Il Mashup Support crea automaticamente un collegamento tra l'*input component* del widget all'interno del quale è stato registrato un evento di tipo "*copy*" (in questo caso Wid_i) e l'*output component* del widget dove ha rilevato un evento di tipo "*paste*" (widget Wid_j). Viene conservato il collegamento tra *input component* e *output component* interno al widget Wid_i.
Dopo che la piattaforma ha creato le connessioni tra i widget, l'aspetto con cui essi vengono presentati all'utente nell'ambiente grafico cambia, infatti vengono nascosti gli *input components* dei widget all'interno dei quali è stato rilevato l'evento di tipo "*paste*".
Nelle seguenti immagini viene mostrata di azioni da effettuare per creare una connessione tra due widget:

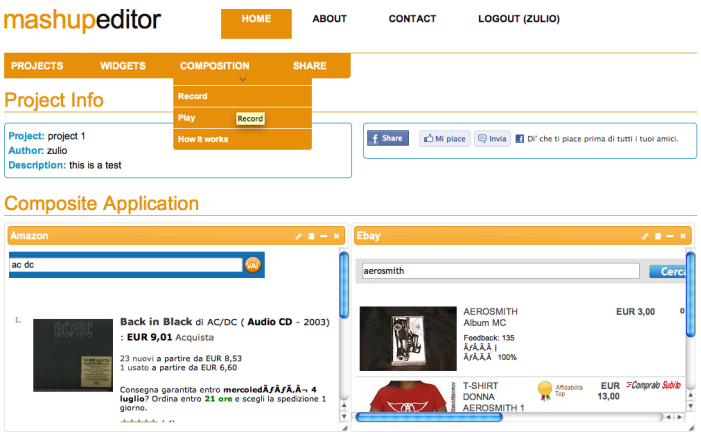

Figura 4.3: Attivazione della modalità di registrazione

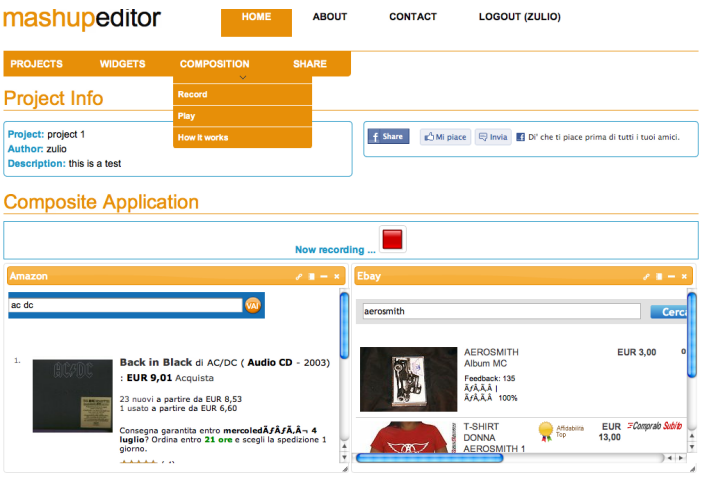

Figura 4.4: Schermata dell'editor in modalità registrazione

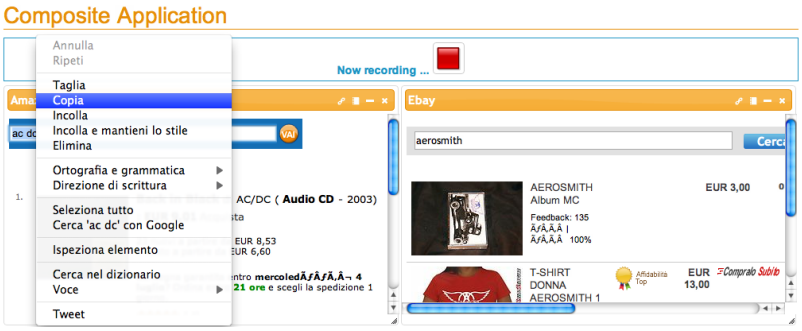

Figura 4.5: Copia del contenuto del text field del widget sorgente

Figura 4.6: Il contenuto degli appunti viene incollato all'interno del text field del widget destinazione

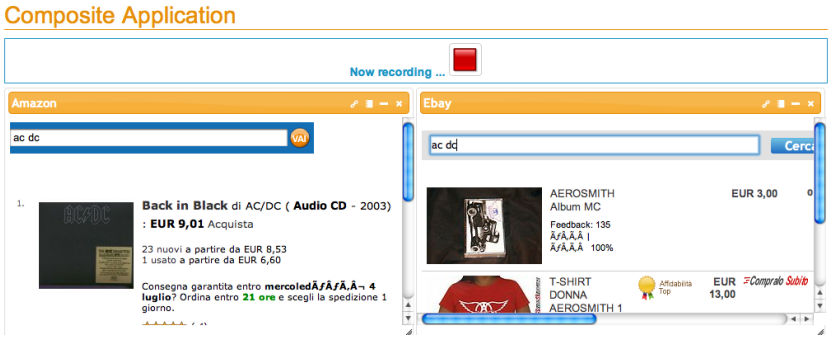

Figura 4.7: Stop alla registrazione

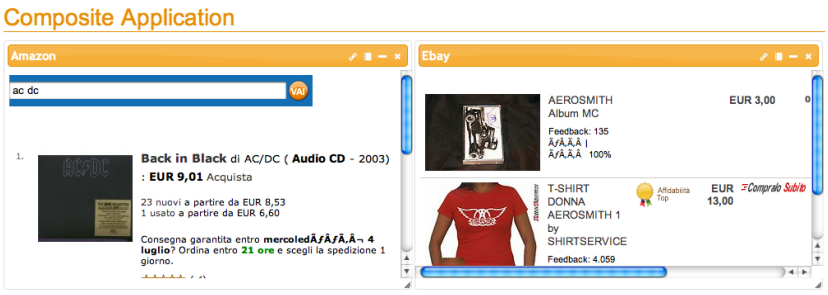

Figura 4.8: La piattaforma mostra il solo il text field del widget sorgente

Una volta effettuata la connessione tra i widget, basta aggiornare il contenuto di un text field e il Mashup Support procederà ad aggiornare il contenuto di tutti i widget collegati.

Le figure 4.9 e 4.10 mostrano il risultato dell'esecuzione del mashup:

Figura 4.9: Contenuto del mashup prima dell'esecuzione

Figura 4.10: Contenuto del mashup dopo l'esecuzione

4.4 Sistema di visualizzazione delle connessioni

Un altro meccanismo aggiunto in questa fase riguarda la visualizzazione delle connessioni presenti nel mashup. Basta attivare la visualizzazione delle connessioni, e l'ambiente grafico evidenzierà le connessioni presenti tra i widget. Durante la fase di visualizzazione vengono mostrate i collegamenti creati tra i vari elementi di input che l'utente ha creato durante la fase di registrazione, per far ciò l'ambiente grafico farà apparire tutti gli *input components* di tutti i widget.

Gli elementi di input collegati tra loro vengono evidenziati modificando il colore di sfondo e il colore del bordo in verde, delle frecce rosse indicano quali sono i collegamenti effettuati. Premendo il tasto stop si disattiva la visualizzazione delle connessioni.

Le seguenti immagini mostrano la visualizzazione delle connessioni presenti:

Figura 4.11: Attivazione della modalità di visualizzazione delle connessioni

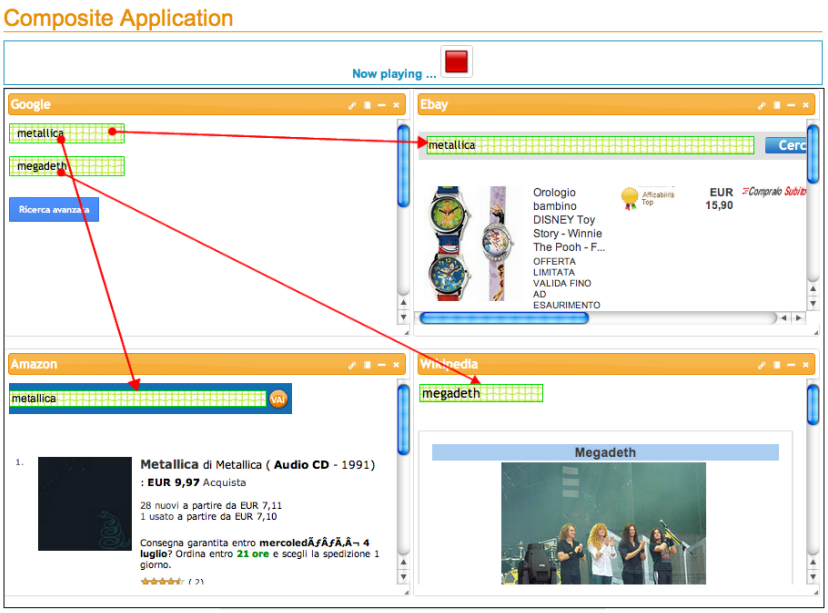

Figura 4.12: Connessioni presenti all'interno del mashup

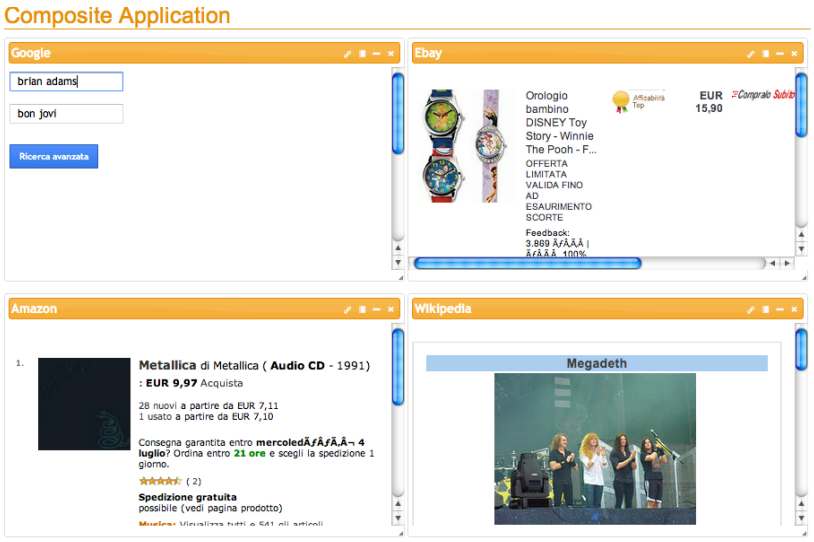

Figura 4.13: Stop della modalità di visualizzazione

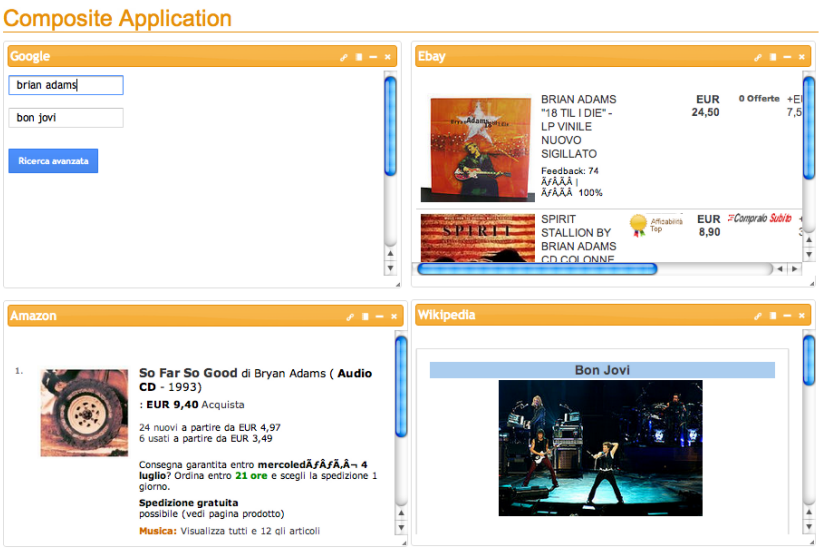

Figura 4.14: Aggiornamento del contenuto del mashup

4.5 Gestione degli utenti

In questa fase dello sviluppo è stato introdotto il supporto per gestire l'accesso contemporaneo di più utenti alla piattaforma. É stato inoltre introdotto un repository locale dove ogni utente può salvare e conservare i propri mashup creati. Per utilizzare la piattaforma ora è necessario registrarsi ed autenticarsi, per memorizzare le informazioni degli utenti è stato introdotto un database che verrà utilizzato inoltre per la gestione del repository condiviso, come vedremo nel prossimo capitolo.

L'introduzione della gestione degli utenti ha portato al cambiamento dell'aspetto dell'EUD Environment, infatti è stata inserita una barra dei menù dove si trovano tutti i comandi delle nuove funzionalità sviluppate.

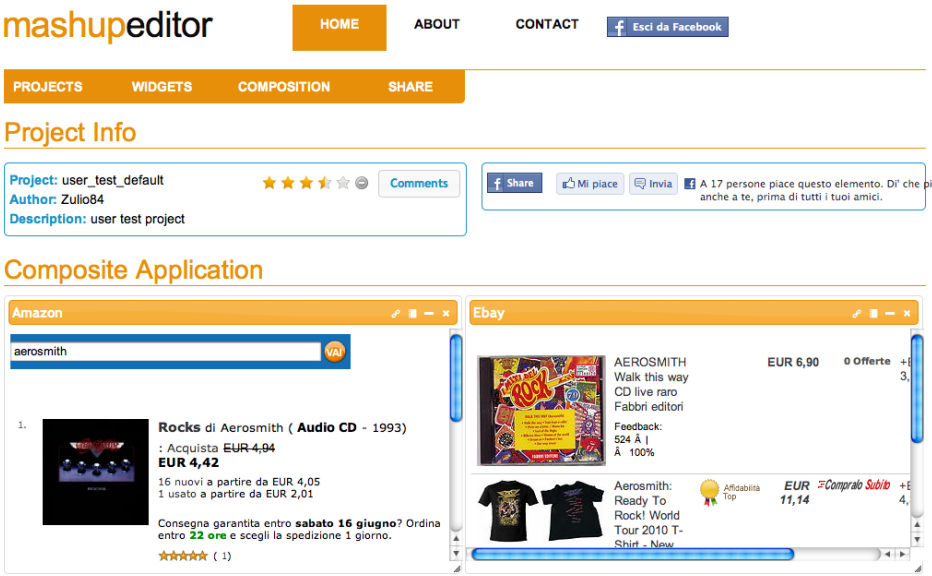

Figura 4.15: Nuova veste grafica del MashupEditor

4.6 Salvataggio e caricamento dei mashup dell'utente

4.6.1 Libreria dei widget

Quando un mashup viene creato all'interno dell'ambiente grafico, appare all'utente come un insieme di widget, all'interno d'ognuno ci sono gli elementi selezionati dall'applicazione originale, che conservano il loro aspetto e le loro funzionalità.
La componente base è dunque il widget, le informazioni necessarie per definirlo sono le seguenti:

- *Titolo*: indica il titolo del widget, viene scelto dall'utente in fase di creazione;
- *Autore*: username del creatore del widget;
- *Data creazione*: data di creazione;

- *Data ultima modifica*: data dell'ultima modifica;
- *URL originale*: si memorizza l'URL originale che ha creato l'output component, viene utilizzata durante la fase di aggiornamento del contenuto di un widget, aggiornando il valore dei parametri utilizzati;
- *Input Component*: si conserva il file HTML con la selezione effettuata dall'utente e la URL al foglio di stile CSS relativo agli elementi presenti nella selezione;
- *Output Component*: simile all'input component si memorizza il file HTML contente gli elementi selezionati nella pagina dei risultati e la URL del CSS di tali elementi;
- *Elenco dei Tag*: elenco dei tag che sono associati ad un widget, sono stati inseriti per permettere la ricerca di widget nel repository attraverso delle parole chiave;
- *Elenco dei link*: elenco degli eventuali link presenti nell'output component;
- *Elenco dei parametri di input*: elenco dei parametri prelevati dall'analisi automatica della richiesta HTTP, in caso di GET sono stati prelevati dalla query string, in caso di POST dalla post content, per ognuno di essi viene salvato l'id che ha all'interno dell'applicazione originale, il suo valore ed un flag che indica se è stato inserito direttamente dall'utente oppure se è un parametro generato automaticamente dall'applicazione originale;
- *Elenco dei parametri collegati*: in questo elenco vengono salvati i collegamenti tra i widget, quando un utente crea un collegamento, viene creato all'interno della struttura dati del widget sorgente una entry per questo elenco; si memorizza l'id del widget destinazione e l'id dei parametri di input collegati.

Per poter salvare i widget è stata creata una *Libreria* all'interno della quale l'utente può aggiungere i propri widget creati tramite la navigazione sul Web. L'utente può caricare direttamente dalla libreria i suoi widget, inserendoli all'interno del mashup che sta creando con un notevole risparmio di tempo rispetto alla creazione originale.
La libreria è a disposizione di ogni utente e lo spazio sul disco necessario per memorizzare i widget viene creato al momento della registrazione dell'utente alla piattaforma, i widget salvati all'interno della libreria non sono visibili agli altri utenti della piattaforma ma sono strettamente personali.
Per ogni widget vengono creati un file XML, contente il *descrittore* del widget, ed una cartella che conterrà il file HTML e CSS necessari per la definizione dell'*input component* e dell'*output component*.
Quando un utente vuole aggiungere un widget alla propria libreria seleziona il comando presente sulla barra superiore del widget. Viene visualizzata una schermata nella quale inserire le informazioni per salvare il proprio widget.
Le seguenti figure mostrano le azioni da esegui per aggiungere un widget alla propria libreria:

Figura 4.16: Avvio il salvataggio del widget nella libreria

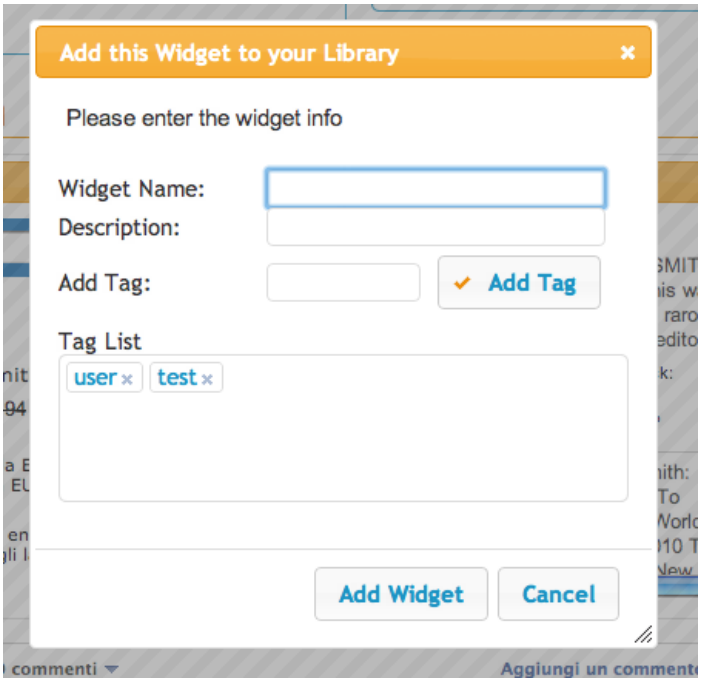

Figura 4.17: Finestra di dialog dove inserire le informazioni relative al widget

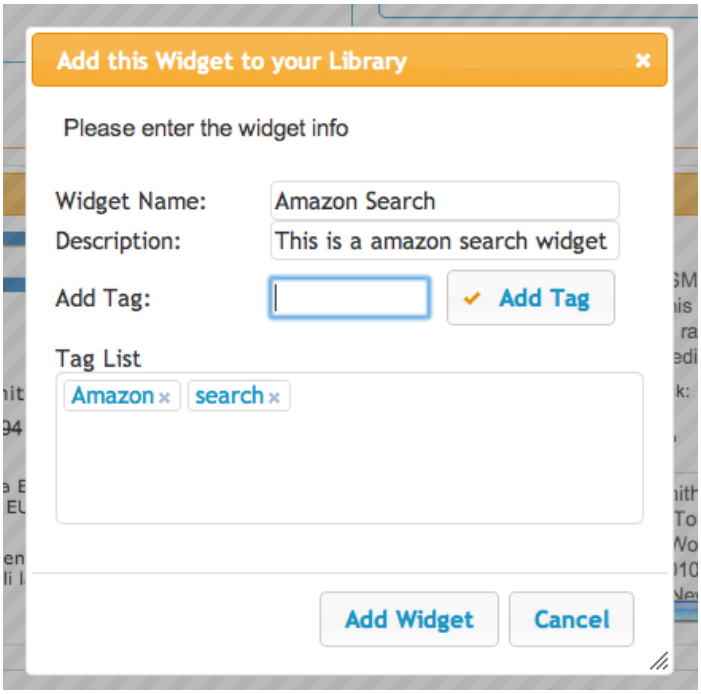

Figura 4.18: Informazioni riguardanti il widget

Le seguenti immagini mostrano come caricare un widget dalla propria libreria:

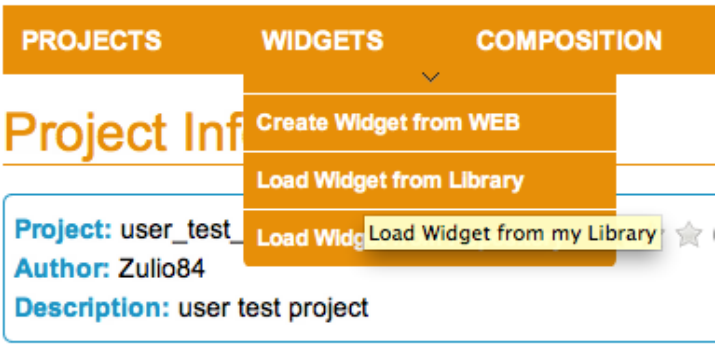

Figura 4.19: Comando per il caricamento del widget dalla libreria

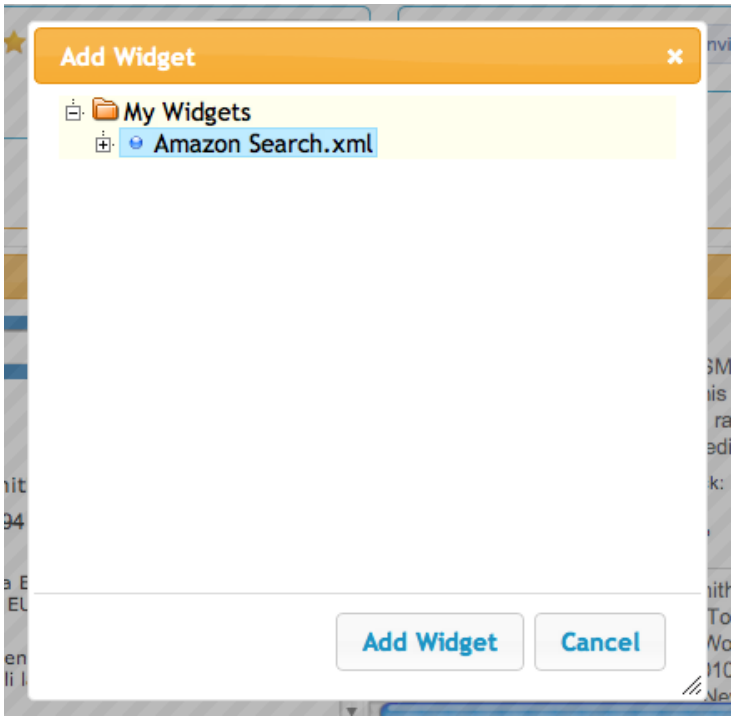

Figura 4.20: Elenco dei widget presenti nella libreria

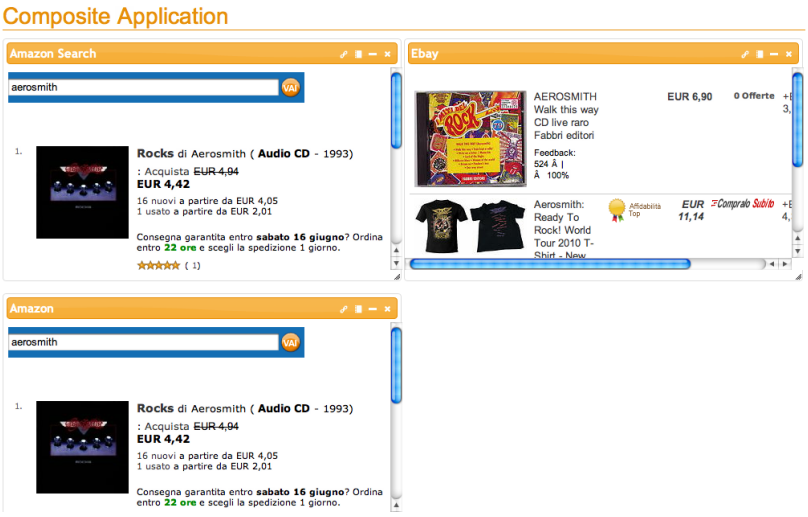

Figura 4.21: Il nuovo widget viene inserito all'interno dell'ambiente di composizione

4.6.2 Gestione dei mashup

L'ultima modifica introdotta in questa fase della tesi riguarda la gestione dei mashup. È stata implementato un modello di gestione dei mashup simile a quello dei widget mostrato nel precedente paragrafo. // Al momento della registrazione sulla piattaforma viene creata l'area di lavoro per l'utente, all'interno della quale vengono memorizzati i progetti di mashup che l'utente crea e poi salva.
Come nel caso dei widget sono state scelte delle informazioni da memorizzare:

- *Nome*: nome del mashup;
- *Descrizione*: descrizione del mashup;
- *Autore*: autore del mashup;
- *Id*: id univoco nel sistema, viene assegnato al momento della creazione;
- *Data Creazione*: data di creazione del mashup;
- *Data ultima modifica*: data dell'ultima modifica effettuata;
- *Elenco dei Tag*: elenco dei tag relativi al mashup, vengono utilizzati per le ricerche nel repository;
- *Elenco dei widget*: elenco dei descrittori dei widget che sono contenuti all'interno del progetto.

Il formato scelto per il salvataggio è l'XML. Il descrittore del mashup include al suo interno anche tutti i descrittori dei widget presenti, mentre viene creata una struttura di directory per salvare i file HTML e CSS che compongono i widget. Lo schema che definisce il formato dei file XML di salvataggio è stato creato definendo uno schema XSD[1] visibile nell'appendice A.
In maniera analoga a quanto accade con i widget, l'area di lavoro locale permette di salvare i progetti che sono visibili ed utilizzabili solo dall'utente

[1]XML Schema Definition

proprietario. Quando viene salvato un mashup, oppure un widget, il Mashup Support effettua una copia della struttura dei file necessari per il salvataggio direttamente all'interno dell'area di lavoro dell'utente.
Viceversa in caso di caricamento avviene il processo opposto, il Mashup Support si occupa di prelevare la struttura dei file dall'area di lavoro locale e prepararla per l'esecuzione.
Il Mashup Support è stato modificato, in maniera analoga all'ambiente grafico, per poter funzionare in modalità multiutente infatti, ogni volta che un utente effettua il login alla piattaforma il Mashup Support gli riserva uno spazio di lavoro al suo interno.

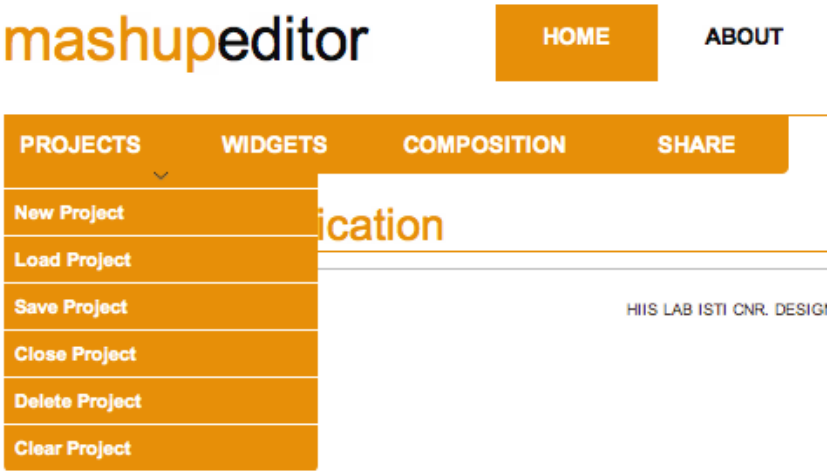

Figura 4.22: Elenco dei comandi per la gestione di un mashup

All'interno dell'ambiente grafico i mashup sono ora definiti come Mashup Projects, l'utente può creare e controllare i propri progetti tramite i comandi presenti nel menù Projects (figura 4.22):

- *New*: lancia la creazione di un nuovo progetto mashup (figure 4.23 e 4.24);
- *Load*: apre la finestra da dove caricare un progetto salvato in precedenza (figura 4.25);

- *Save*: salvataggio del mashup;
- *Close*: chiude il progetto aperto, verrà svuotato lo spazio di lavoro dell'utente all'interno del Mashup Support;
- *Delete*: elimina il mashup aperto nell'editor, cancellando tutti i file relativi ad esso;
- *Clear*: svuota il progetto, eliminando i widget presenti.

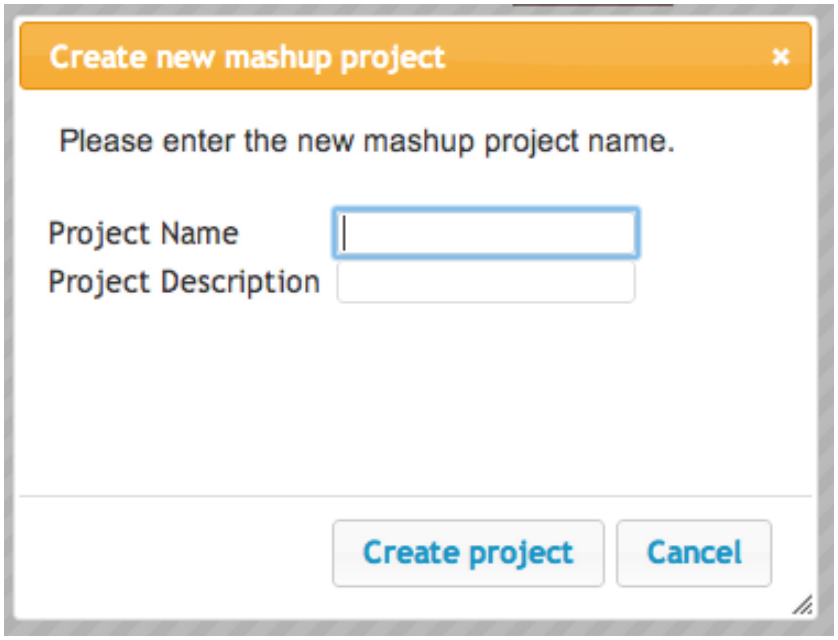

Figura 4.23: Finestra per la creazione di un nuovo progetto mashup

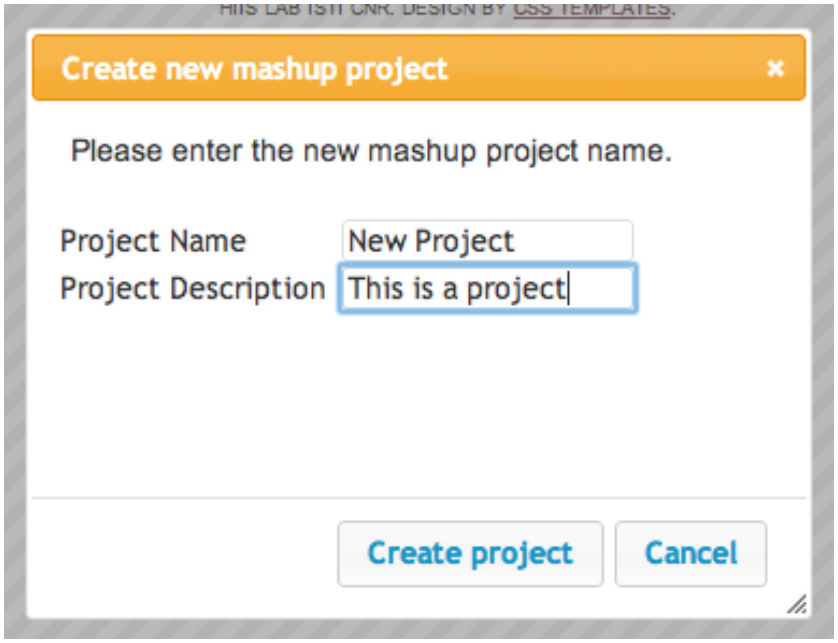

Figura 4.24: Inserimento delle informazioni sul nuovo progetto

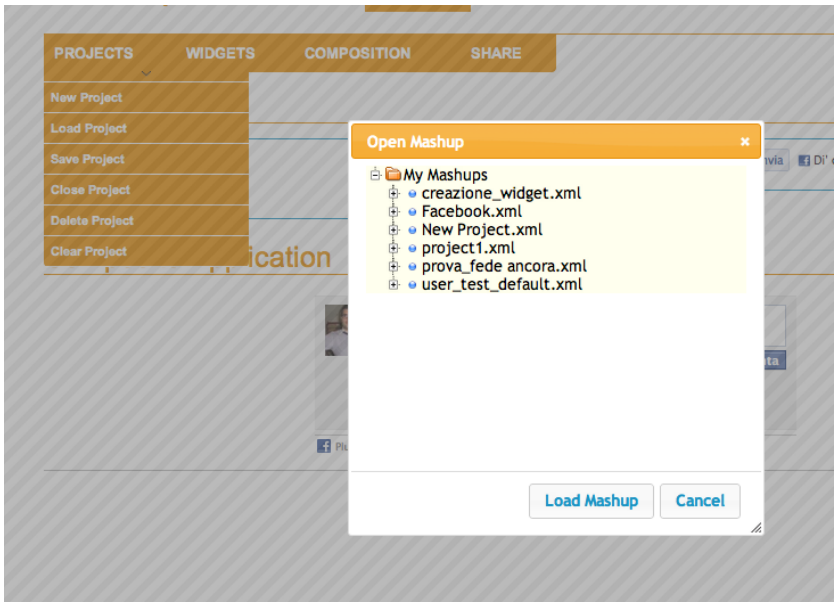

Figura 4.25: Schermata che mostra i progetti dell'utente

4.7 Implementazione

Con riferimento alla struttura della piattaforma presentata nel paragrafo 3.1, per sviluppare le funzionalità presentate all'interno del capitolo ho lavorato inizialmente sul lato server della piattaforma, modificando il Mashup Support. Questa componente del sistema consiste in una Servlet Java che mette a disposizione dell'ambiente grafico il supporto per la gestione dei mashup.
Via via che le funzionalità venivano aggiunte sul lato server, ho provveduto a modificare anche il lato client, inserendo all'interno dell'ambiente grafico i comandi per poterle utilizzare.
Il codice relativo al Mashup Support analizzato all'inizio del percorso di tesi comprendeva una serie di classi Java per il supporto dei widget e delle connessioni tra di essi, queste classi venivano utilizzate all'interno della Servlet per gestire dei mashup che non potevano però essere salvati e ricaricati.
Durante la prima parte dell'implementazione mi sono è concentrato sulla definizione delle classi per la gestione dei mashup e del supporto per il loro salvataggio. All'interno del codice ho introdotto un package chiamato *entities* che raggruppa le seguenti classi:

- *MashupProjectDescriptor.java*: è la rappresentazione nel sistema di un mashup project, all'interno di questa classe si trovano tutte le informazioni descritte nel paragrafo sulla gestione dei mashup;
- *MashupDescriptor.java*: rappresenta il widget all'interno del sistema, contiene le informazioni descritte nel paragrafo sulla libreria dei widget, ogni *MashupProjectDescriptor* contiene al suo interno una lista di oggetti di questo tipo;
- *ParameterDescriptor.java*: questa classe viene utilizzata per rappresentare le informazioni relative agli elementi HTML di input che sono stati selezionati dall'utente durante la navigazione Web, per ognuno di essi si memorizzano *id, label, name, valore* e *tipo*;

- *SendInputDescriptor.java*: ogni oggetto di questa classe rappresenta una connessione tra due widget, ogni *MashupDescriptor* ha una lista di oggetti di questo tipo, vengono inseriti all'interno della lista momento della creazione automatica delle connessioni. Sono presenti all'interno di questi oggetti le informazioni relative agli *id* degli elementi di input utilizzati per connettere due widget, e il riferimento al widget collegato.

- *ActionRecord.java*: all'interno di questa classe vengono memorizzare le informazioni relative ad un'azione compiuta dall'utente durante la fase di registrazione, all'interno troviamo infatti le informazioni relative al widget utilizzato, l'id dell'elemento di input utilizzata, il tipo di evento rilevato ed il valore dell'elemento di input.

Dopo l'introduzione della classe *MashupProjectDescriptor.java* la gestione di un mashup avviene ora attraverso l'utilizzo di oggetti di questo tipo, era necessario però consentire il salvataggio su file dei mashup, ho quindi definito tramite uno schema XSD, le informazioni necessarie per il salvataggio.
Il passaggio successivo è stato il procedimento di binding tra la rappresentazione XML dello schema e le corrispondenti classi Java che rappresentano lo schema del salvataggio. Utilizzando NetBeans e le librerie JAXB[2], ho generato le classi Java necessarie che sono state messe all'interno di un package dedicato di nome *it.cnr.isti.giove.mashupenvironment*.
Per il caricamento di un file XML definito secondo lo schema del salvataggio si utilizza l'operazione di *unmarshalling*, questa operazione genera gli oggetti Java corrispondenti al contenuto del file XML, in questo caso per un file contente un mashup verrà creato un oggetto di tipo
it.cnr.isti.giove.mashupenvironment.MashupProjectDescriptor.java.
Per il suo utilizzo all'interno della piattaforma di mashup, c'è bisogno di un ulteriore passaggio bisogna infatti mapparlo all'interno di un oggetto di tipo *entities.MashupProjectDescriptor.java*, questo avviene attraverso l'utilizzo di una classe di supporto chiamata *EntitiesWrapper.java* che realizza la conver-

[2]Java Architecture for XML Binding

sione.
Il procedimento di salvataggio è esattamente l'inverso, attraverso la classe *EntitiesWrapper* si passa dal descrittore di mashup per la piattaforma, a quello che rappresenta il file XML, infine tramite l'operazione di *marshalling* si genera il contenuto del file XML di salvataggio.
Lo stesso tipo di meccanismo è stato implementato per la gestione del salvataggio dei widget per la libreria personale dell'utente, in questo caso quando un utente carica un widget salvato in precedenza l'operazione di *unmarshalling* genera un oggetto di tipo:
it.cnr.isti.giove.mashupenvironment.MashupDescriptor.java, l'*EntitiesWrapper* lo converte poi in un oggetto di tipo *entities.MashupDescriptor.java*, che è utilizzato all'interno *entities.MashupProjectDescriptor.java*.
Una volta terminata l'implementazione del caricamento e salvataggio dei mashup e dei widget nel lato server, ho inserito all'interno del lato client gli strumenti per accedere alle operazioni implementate, ho inserito i comandi all'interno di un menù realizzato con JqueryUi (figura 4.15).
Per consentire la gestione di più utenti nel sistema contemporaneamteè stata inserita nel sistema la definizione di utente tramite la classe *MashupUser.java*. Le informazioni che sono presenti all'interno della classe utente sono le seguenti:

- *username*: username con cui un utente è registrato nel sistema;
- *myOpenProjects*: lista dei progetti mashup che l'utente ha in esecuzione nell'editor;
- *projectsBaseDir*: path della sua area di lavoro;
- *widgetBaseDir*: path della sua libreria personale.

Quando un utente effettua il login le sue credenziali vengono controllate all'interno del database, se l'esito dell'autenticazione è positivo allora viene creato un oggetto relativo all'utente, che la Servlet memorizza all'interno

della sessione HTTP. L'oggetto utente viene rimosso dopo il logout oppure quando la sessione va in timeout.

L'implementazione del sistema delle connessioni tramite la registrazione è stata implementata nel seguente modo:

1. *Lato client*: l'utente avvia la registrazione delle sue azioni tramite il comando "*record*" (figura 4.3).

   Lato server: la Servlet riceve il comando di registrazione e crea una nuova lista di oggetti di tipo *ActionRecord.java*.

```
session.setAttribute("recording", true);
if (session.getAttribute("actionList") == null) {
  session.setAttribute("actionList", new ArrayList<ActionRecord>());
  } else {
   List<ActionRecord> actionList =
 (List<ActionRecord>) session.getAttribute("actionList");
              actionList.clear();
         }

  // svuota le connessioni precedenti tra i widget
  if (elements != null) {
        for (ArrayList<MashupDescriptor> arrayList : elements) {
           if (arrayList != null) {
               for (MashupDescriptor mashupDescriptor : arrayList) {
                   mashupDescriptor.getSendInputs().clear();
                   // reset dei parametri
                mashupDescriptor.updateInputParameters();
                 }
            }
         }
     }
```

2. *Lato client*: l'utente compie delle azioni all'interno dell'ambiente grafico (figure 4.4 e 4.5), ogni azione effettuata viene inviata al server tramite una chiamata AJAX.

   Lato server: il server riceve le chiamate AJAX contenenti le descrizioni delle azioni, per ognuna di esse crea un oggetto di tipo *ActionRecord.java* e lo aggiunge alla lista delle azioni registrate:

```
List<ActionRecord> actionList =
     (List<ActionRecord>) session.getAttribute("actionList");
              ActionRecord record = new ActionRecord(
                      request.getParameter("widget_id"),
                      request.getParameter("input_id"),
                      request.getParameter("event"),
                      request.getParameter("value"));

              actionList.add(record);
```

3. *Lato client*: l'utente ferma la registrazione (figura 4.7).

   Lato server: il server riceve il comando di stop ed elabora le operazioni registrate, per ogni azione registrata controlla il tipo di evento che l'ha generata, se si tratta di un evento di tipo *cut* o *copy* il widget all'interno del quale è stata registrata tale azione viene utilizzato come *sender* e memorizzato, quando viene individuato un evento di tipo *paste* il widget corrispondente a tale evento viene utilizzato come *receiver*.
   All'interno del widget sender si crea un nuovo oggetto di tipo *SendInputDescriptor.java* che rappresenta la connessione tra il sender e il receiver. l'oggetto connessione viene popolato inserendo all'interno gli id degli elementi di input da connettere, e il riferimento al widget receiver.

   La connessione viene salvata all'interno del widget sender.

```
 ArrayList<ArrayList<MashupDescriptor>>
 elements = currentProject.getElements();

   ActionRecord lastCopy = null;
  if (session.getAttribute("actionList") == null) {
  return;
   }
 List<ActionRecord> records =
(List<ActionRecord>) session.getAttribute("actionList");
 for (ActionRecord r : records) {
 if (r.getEvent().equals("copy") || r.getEvent().equals("cut")) {
 lastCopy = r;
  } else if (r.getEvent().equals("paste")) {
  // add the connection here
 if (lastCopy != null) {
 // Widget dal quale ho copiato/tagliato la parola
MashupDescriptor sender = searchWidget(elements, lastCopy.getWidgetId());
 // Widget dove ho incollato
 MashupDescriptor receiver = searchWidget(elements, r.getWidgetId());
  // genero una connessione tra i due widget
 SendInputDescriptor connection = new SendInputDescriptor();
 // inserisco l'id della input da dove prelevo il valore
 connection.setInputKey(lastCopy.getInputId());
 // inserisco l'id dell'input dove inserisco il valore
 connection.setOutputKey(r.getInputId());
 // inserisco il widget destinatario
connection.setWidget(receiver);
  // creo una connessione tra il sender e il destinatario
 sender.getSendInputs().add(connection);
     }
   }
 }
```

4. *Lato Client*: L'ambiente grafico si aggiorna modificando la visualizzazione dei widget di tipo *receiver*, infatti per ognuno di essi viene nascosto l'*input component* (figura 4.8).

Capitolo 5

Supporto alla creazione di una comunità di utenti

La seconda parte delle modifiche effettuate sulla piattaforma ha portato alla creazione di una *community*, attraverso la quale gli utenti possono condividere, commentare e valutare le proprie esperienze e soluzioni nella creazione di mashup.

5.1 Implementazione del repository

La prima fase dell'implementazione si è concentrata sulla definizione di un database di supporto alla piattaforma. Il database utilizzato per la gestione degli utenti della piattaforma, visto nel precedente capitolo, è stato ridefinito per memorizzare le informazioni relative ai mashup e ai widget che si vogliono condividere con gli altri utenti.
Il database, la cui struttura è visibile nella figura 5.1, è stato chiamato "*mashup_rep*" e viene gestito da un DBMS[1] MySQL.

[1]Database management system

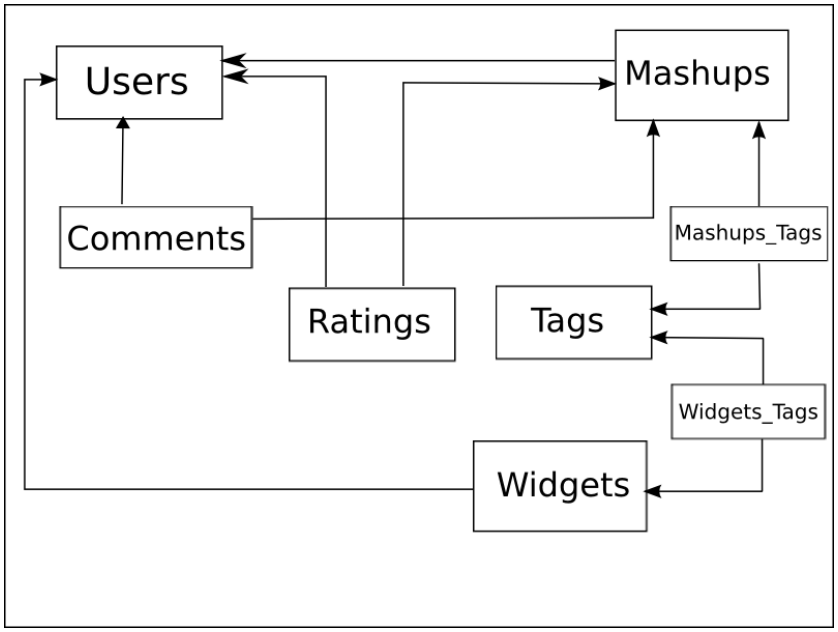

Figura 5.1: Struttura del database del repository

Le informazioni che vengono salvate sono organizzate per favorire la ricerca dei mashup e dei widget che vengono condivisi dagli utenti. Quando viene effettuata una condivisione, di un mashup o di un widget, le loro informazioni, che sono state presentate nel capitolo 4 e che vengono salvate all'interno dei file XML, vengo replicate e inserite all'interno del database.
Gli utenti della piattaforma possono effettuare delle ricerche in base a:

- *Nome dell'autore* del widget o del mashup;
- *Nome* del widget o del mashup;
- *Tag* inseriti per identificare il widget o il mashup.

Il database è affiancato da un file system all'interno del quale vengono memorizzati i file HTML che servono per comporre i widget ed i mashup. All'interno dell'ambiente grafico l'utente può accedere al repository dei mashup utilizzando i comandi del menù "*SHARE*" (figura 5.2).

Figura 5.2: Comandi del menù Share

5.2 Condivisione di un mashup

L'utente che vuole effettuare la condivisione di un mashup project all'interno del repository deve selezionare il comando di condivisione all'interno del menù ed inserire le informazioni sul progetto che vuole condividere (figura 5.3). Una volta effettuata la condivisione, il progetto dell'utente sarà disponibile agli altri utenti che potranno ricercarlo secondo uno dei metodi illustrati nel precedente paragrafo.

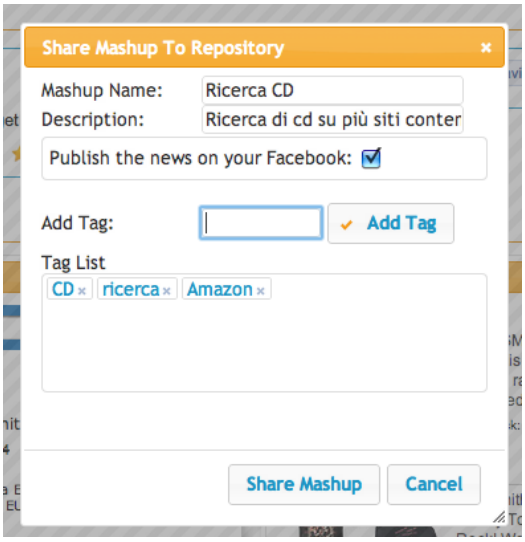

Figura 5.3: Finestra per l'inserimento delle informazioni del mashup da condividere

Quando un progetto viene condiviso ottiene un ID univoco all'interno del sistema, tale ID lo identifica e lo distingue dagli altri progetti. L'ID di un mashup project viene assegnato dal Mashup Support al momento della creazione da parte dell'utente e non potrà più essere modificato. Un progetto condiviso possiede una URL unica all'interno del MashupEditor, tale URL è della forma: "*MASHUPEDITOR_URL/CommandServlet&p_id=ID*", se un utente accede alla piattaforma inserendo tale URL nel browser (figura 5.4) l'ambiente grafico caricherà automaticamente il progetto corrispondente alla URL inserita:

Figura 5.4: Esempio di URL univoca nel sistema

Quando un progetto viene condiviso, il Mashup Support crea una copia dei file che definiscono un mashup project all'interno dell'area di lavoro locale dell'utente e li salva all'interno del repository condiviso in una directory che ha come nome l'ID univoco del progetto. Da questo momento in poi ogni modifica che l'utente effettua sul suo progetto in locale viene riportata automaticamente anche all'interno del repository.

5.3 Caricamento di un mashup

Il caricamento di un progetto dal repository avviene ricercando all'interno del database la parola chiave che l'utente inserisce all'interno della form di ricerca (figura 5.5). La ricerca nel database verrà effettuata in base alla tipologia di ricerca che l'utente sceglie tra quelle disponibili (figura 5.6), infine quando viene selezionato un risultato della ricerca la piattaforma caricherà il mashup scelto (figure 5.7 e 5.8).

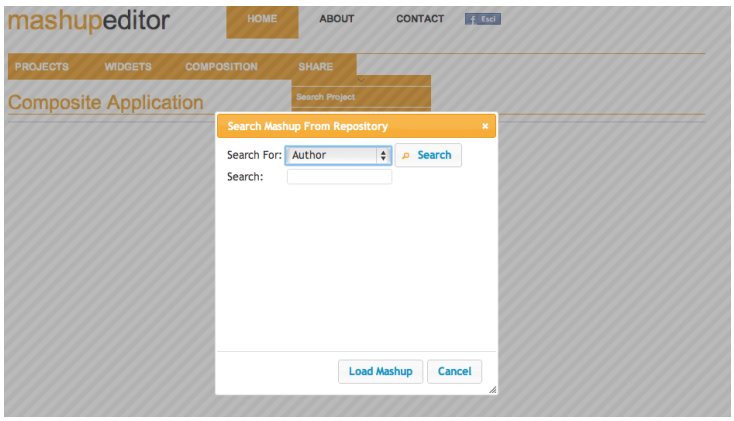

Figura 5.5: Finestra per la ricerca nel repository

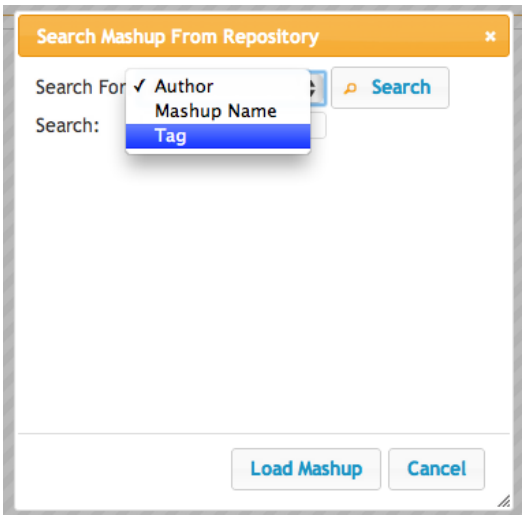

Figura 5.6: Scelta della tipologia di ricerca

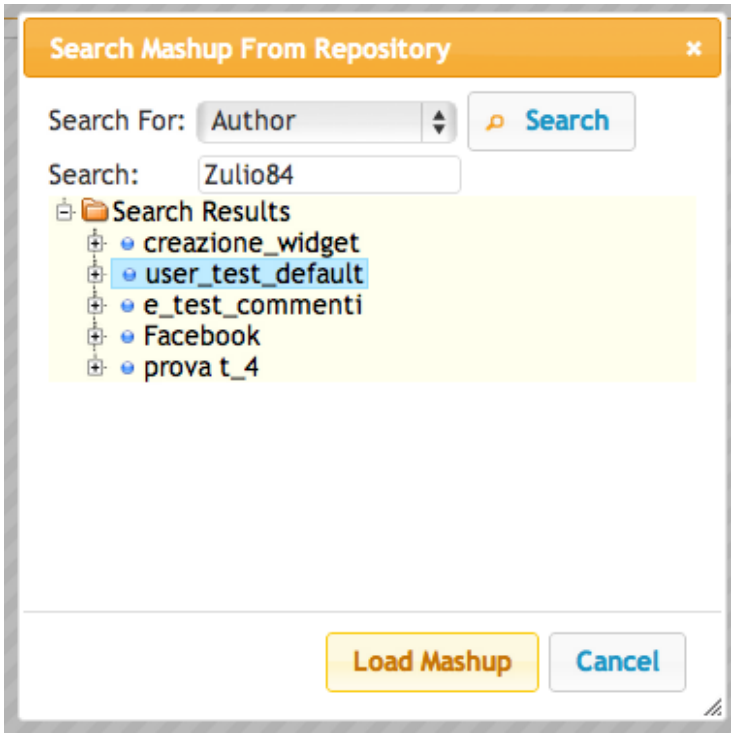

Figura 5.7: Selezione del mashup desiderato

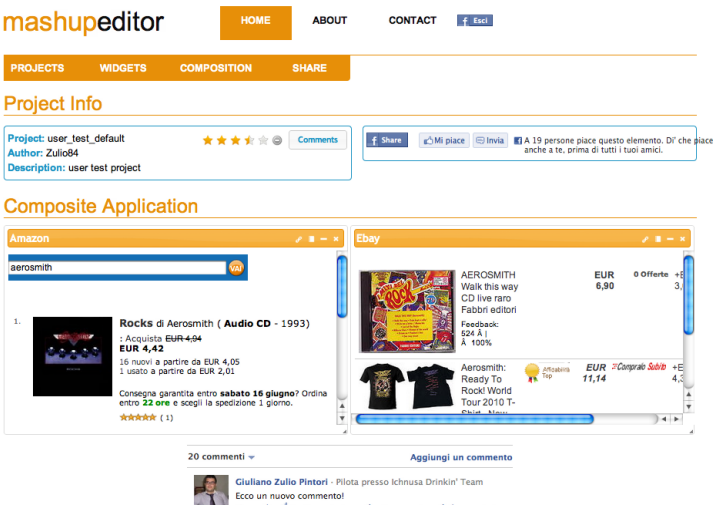

Figura 5.8: Caricamento del mashup selezionato

Quando un utente carica un progetto dal repository, il Mashup Support crea una copia di tale mashup nell'area di lavoro locale dell'utente.

Nel momento in cui vengono effettuate delle modifiche al mashup la piattaforma effettua il salvataggio in locale delle modifiche. Se l'utente è anche il proprietario del mashup tali modifiche vengono riportate in automatico anche sulla copia presente nel repository, se invece l'utente non è il possessore del mashup tali modifiche rimarranno persistenti solo sulla sua copia locale. Quando un utente che modifica un progetto di cui non è il proprietario vuole condividere le sue modifiche con gli altri utenti, può farlo effettuando una nuova condivisione del mashup.
Al mashup verrà associato un nuovo ID univoco all'interno della piattaforma e l'utente ne diverrà il proprietario.

5.4 Votazione e Commenti

Gli utenti di una community interagiscono tra di loro votando e commentando i contenuti dei siti ai quali sono iscritti, il MashupEditor è stato anche esso dotato di un sistema per consentire agli utenti di votare e commentare i mashup che vengono condivisi.
Quando un utente crea un mashup, i comandi per la votazione e i commenti sono nascosti, appena viene effettuata la condivisione del mashup nel repository essi appaiono nella zona delle informazioni del progetto (figura 5.9).

Figura 5.9: Visualizzazione delle funzioni voto e commento

Il database è stato ampliato inserendo le tabelle "*Comments*" e "*Ratings*" per memorizzare le opinioni ed i voti dei vari utenti. Quando un utente seleziona il comando "*Comments*" nell'ambiente grafico, una finestra di dialogo mostra gli ultimi commenti inseriti ordinati a partire dal più recente (figura 5.10).

L'utente può aggiungere il proprio commento inviandolo attraverso la form presente nella parte alta della finestra (figure 5.11 e 5.12).

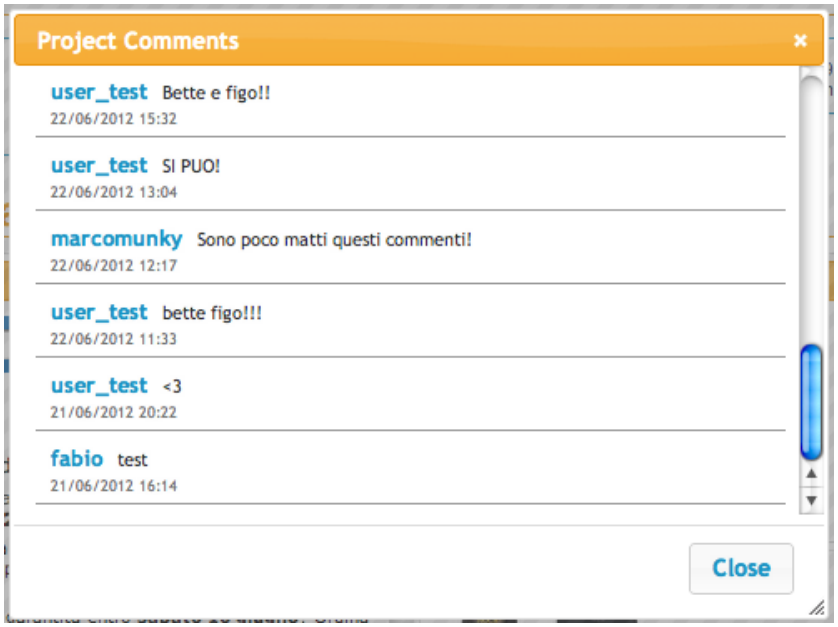

Figura 5.10: Finestra con i commenti precedenti

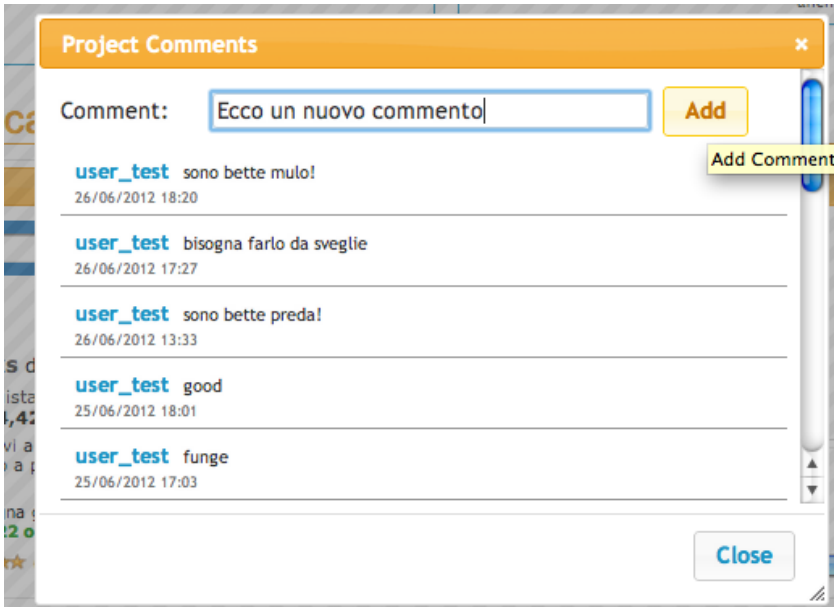

Figura 5.11: Inserimento di un commento 1

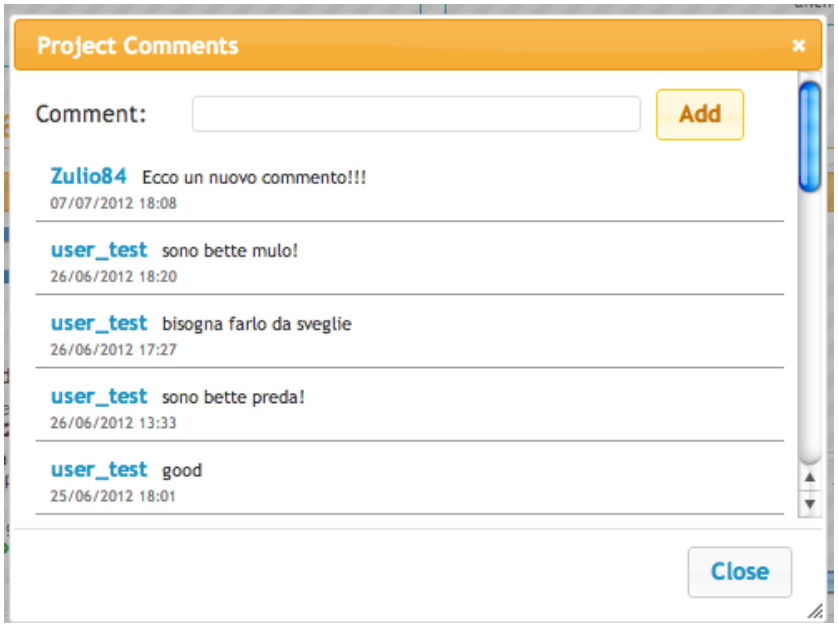

Figura 5.12: Inserimento di un commento 2

Il sistema di voto consente ad un utente di valutare un mashup esprimendo un voto che va da 1 a 5. Ogni utente contribuisce con un singolo voto per la valutazione di un mashup, nel caso in cui voglia cambiare o eliminare il suo voto, può ripetere la votazione ed il voto memorizzato verrà aggiornato con il nuovo valore (figure 5.13, 5.14 e 5.15).
Il voto che viene visualizzato all'interno delle informazioni del progetto è il voto medio del progetto, viene calcolato facendo la media di tutti i voti registrati.

Figura 5.13: Visualizzazione del rating del mashup

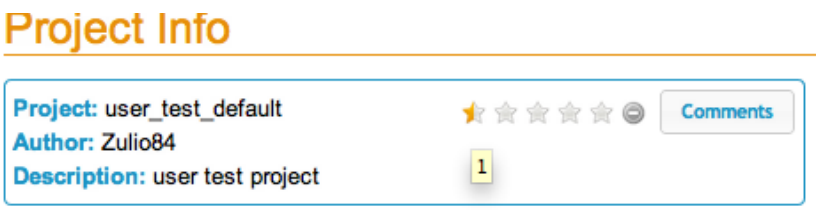

Figura 5.14: Inserimento di un voto

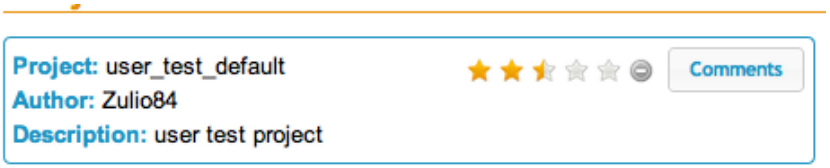

Figura 5.15: Rating del mashup aggiornato dopo l'inserimento del voto

5.5 Condivisione di un widget

L'ultima funzionalità inserita per creare una community riguarda la condivisione dei widget tra gli utenti. Condividere i widget tra gli utenti è importante, perché consente di risparmiare tempo durante la fase di composizione di un mashup, spesso infatti il widget che un utente può volere è già stato creato e condiviso da altri utenti.
Per condividere un widget si utilizza il comando posto sulla barra superiore del widget stesso. L'utente inserisce le informazioni sul widget attraverso una form, ed effettua la condivisione, da questo momento in poi il widget è disponibile per gli altri utenti della piattaforma (figure 5.16 e 5.17).
I file HTML necessari per la definizione di un widget vengono memorizzati in un file system all'interno del server, mentre le informazioni utili per la sua ricerca vengono inserite all'interno del database di supporto.

Figura 5.16: Comando per la condivisione di un widget nel repository

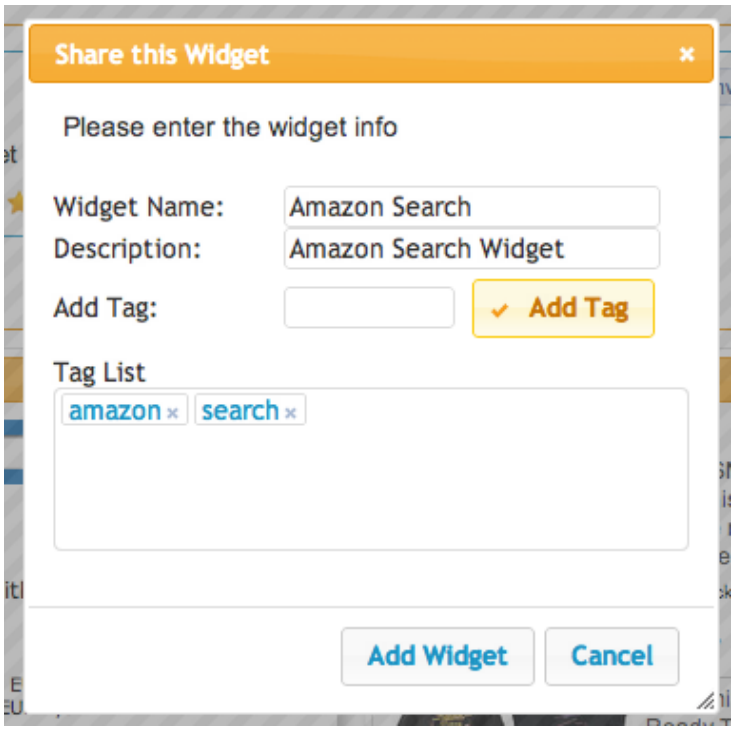

Figura 5.17: Finestra per l'inserimento delle informazioni relative al widget

5.6 Caricamento di un widget

Si può ricercare un widget nel repository tramite la form di ricerca, che viene visualizzata selezionando il comando "*Load Widget from Repository*", nel menù "*Widgets*" (figura 5.18).
Si può effettuare la ricerca di un widget ricercando tre tipi di informazioni:

- *Widget Name*: si ricercano i widget inserendo il nome del widget ricercato;
- *Autore*: si ricercano i widget inserendo l'username del creatore del widget;
- *Tag*: si ricercano i widget in base ai tag che sono stati inseriti in fase di condivisione.

Una volta effettuata la ricerca (figura 5.19) si sceglie il widget tra i risultati, dopodiché il Mashup Support effettua una copia dei file del widget all'interno dell'area di lavoro dell'utente. L'ambiente grafico infine mostra il nuovo widget all'interno della finestra dell'editor, pronto per essere collegato agli altri widget presenti (figura 5.20).

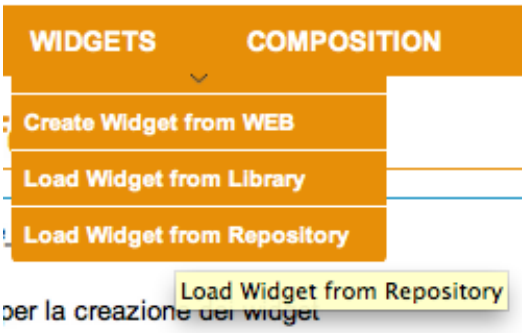

Figura 5.18: Comando per il caricamento di un widget dal repository

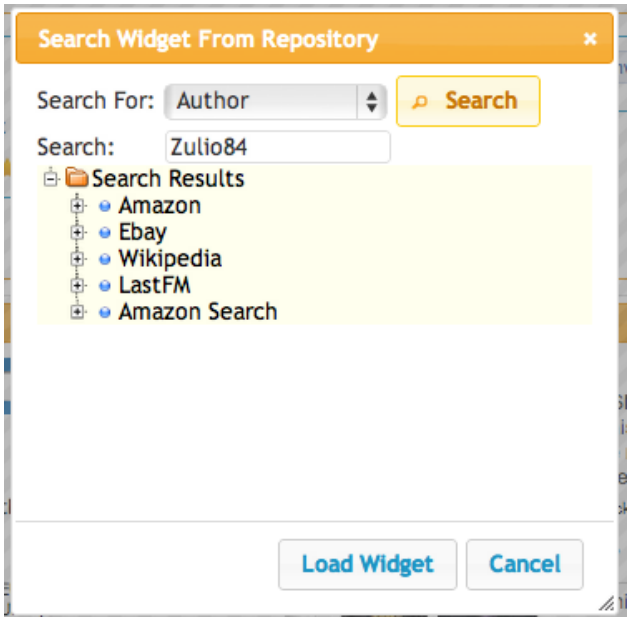

Figura 5.19: Risultati della ricerca di un widget

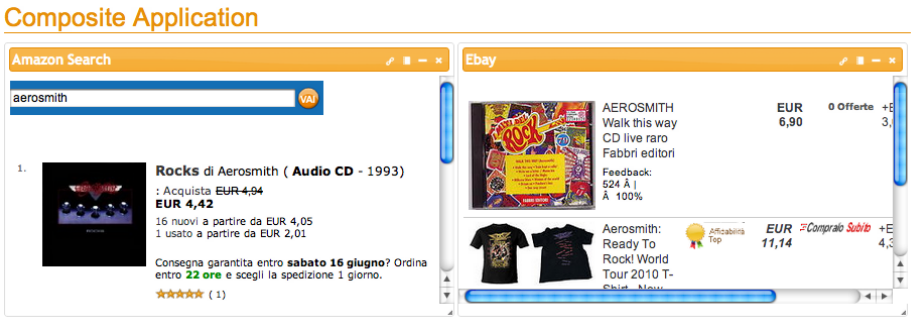

Figura 5.20: Il nuovo widget all'interno dell'editor

5.7 Implementazione

L'implementazione del reposistory è stata fatta lavorando sul componente del sistema di nome *Mashup Repository* (figura 3.1). Il componente è composto da due classi Java che collegano il componente Mashup Support con il database.
Le classi Java che compongono il componente sono:

- *DBManager.java*: Questa classe implementa un Connection Pool per la gestione delle connessioni con il database. Ogni volta che un componente del sistema deve accedere al database richiede una connessione tramite questa classe.

  Il numero delle connessioni che sono disponibili è configurabile all'interno del file *Web.xml* dell'applicazione MashupEditor.

  Codice per la creazione del *DBManager*:

```
public DBManager(String user, String pass, String url,
                                String driver, int cn)
 throws ClassNotFoundException {
        this.LOGIN = user;
        this.PASSWORD = pass;
        this.URL = url;
        this.DRIVER = driver;
        this.CONNECTION_NUMBER = cn;


        Class.forName(DRIVER);

        connections = new Connection[CONNECTION_NUMBER];
        connectionStatus = new ConnectionStatus[CONNECTION_NUMBER];
        for (int i = 0; i < connectionStatus.length; i++) {
            connectionStatus[i] = ConnectionStatus.Uninitialized;
        }
    }
```

  Codice per la richiesta e il rilascio di una connessione:

```
public synchronized Connection getConnection()
 throws SQLException {
        while (true) {
            for (int i = 0; i < connections.length; i++) {
                switch (connectionStatus[i]) {
                    case Uninitialized: {
                        connections[i] =
                        DriverManager.getConnection(URL, LOGIN, PASSWORD);
```

```
                        connectionStatus[i] =
                        ConnectionStatus.Busy;
                        return connections[i];
                    }
                    case Available: {
                        connectionStatus[i] =
                         ConnectionStatus.Busy;
                        return connections[i];
                    }
                }
            }
            try {
                wait();
            } catch (InterruptedException ex) {
                Logger.getLogger(DBManager
                .class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

public synchronized void releaseConnection(Connection toRelease) {
    for (int i = 0; i < connections.length; i++) {
       if (toRelease == connections[i]) {
          connectionStatus[i] = ConnectionStatus.Available;
           notify();
          break;
        }
    }
}
```

- *SQLManager.java*: Questa classe rappresenta l'interfaccia tra le componenti del sistema e il database. Fornisce una serie di funzioni che le componenti del sistema utilizzano per eseguire le operazioni sul database. All'interno di ogni funzione il SQLManager esegue la query SQL corripondente all'operazione richiesta.
  Il seguente codice mostra l'implementazione dell'operazione di *login*:

```
public static int login(Connection con, String username, String password) {
        int toRet = -1;

        if (con == null
                || username == null
                || password == null) {
            return -1;
        }

        PreparedStatement ps = null;
        ResultSet rs = null;
        try {


            StringBuilder sb = new StringBuilder(0);
            sb.append("SELECT").append(SPACE);
            sb.append(Users.user_id).append(SPACE);
```

```
            sb.append(COMMA).append(SPACE);
            sb.append("COUNT(*) AS trovati").append(SPACE);
            sb.append("FROM").append(SPACE);
            sb.append(Users.tableName).append(SPACE);
            sb.append("WHERE").append(SPACE);
            sb.append(Users.username).append(SPACE);
            sb.append(EQUAL).append(SPACE)
            .append(PARAMETER).append(SPACE);
            sb.append("AND").append(SPACE);
            sb.append(Users.password).append(SPACE);
            sb.append(EQUAL).append(SPACE).append(PARAMETER);
            sb.append(SPACE).append("GROUP BY")
            .append(SPACE).append(Users.user_id);

            ps = con.prepareStatement(sb.toString());

            ps.setString(1, username);
            ps.setString(2, password);

            rs = ps.executeQuery();

            while (rs.next()) {
                int res = rs.getInt("trovati");

                if (res != 0) {
                    //restituisco il valore della chiave user_id trovata;
                    toRet = rs.getInt(Users.user_id);
                }
            }
        } catch (SQLException ex) {
            Logger.getLogger(SQLManager.class.getName())
            .log(Level.SEVERE, null, ex);
            toRet = -1;
        } finally {
            if (rs != null) {
                try {
                    rs.close();
                } catch (SQLException ex) {
                    Logger.getLogger(SQLManager.class.getName())
                    .log(Level.SEVERE, null, ex);
                }
            }

            if (ps != null) {
                try {
                    ps.close();
                } catch (SQLException ex) {
                    Logger.getLogger(SQLManager.class.getName())
                    .log(Level.SEVERE, null, ex);
                }
            }
        }
        return toRet;
    }
```

Capitolo 6

Integrazione con Facebook

I social network sono uno strumento tipico del Web 2.0, essi consentono la creazione, condivisione e lo scambio di contenuti prodotti dagli utenti. Una delle principali caratteristiche è data dalla possibilità di stabilire delle relazioni di tipo personale oppure lavorativo. Ogni utente può creare dei contenuti e metterli a disposizione degli altri utenti, tutti gli utenti a cui lui è collegato hanno accesso a tali contenuti.
Ogni utente collegato può usufruire di tali contenuti ed eventualmente può metterli a disposizione dei suoi collegamenti, accrescendo il bacino di utenza del contenuto condiviso, questo principio favorisce dunque la rapida diffusione dei contenuti all'interno dei social network. I social network forniscono inoltre una serie di strumenti per supportare il dialogo tra gli utenti, attraverso tali strumenti gli utenti possono scambiarsi messaggi, commenti o valutazioni sui contenuti.
I contenuti inseriti nella rete sociale dagli utenti che hanno un grande numero di contatti, raggiungono un bacino di utenza molto elevato, è possibile che all'interno di tali gruppi di persone ci siano persone che sono interessate al contenuto condiviso, ed altre che non lo sono. In quest'ottica è stata progettata ed inserita all'interno della piattaforma di mashup l'integrazione con uno dei social network maggior utilizzati dagli utenti: *Facebook*.
L'integrazione con Facebook consente di aumentare il bacino d'utenza della

piattaforma di mashup, gli utenti registrati su Facebook possono utilizzare il MashupEditor per la creazione di mashup e condividere i propri progetti all'interno del social network.
I *social plugin* messi a disposizione da Facebook sono stati inseriti all'interno del'ambiente grafico, gli utenti sviluppano i loro mashup utilizzando un ambiente a loro più familiare in cui trovano alcune delle funzionalità che utilizzano normalmente sul social network come il *Like Button*.
La diffusione dei contenuti della piattaforma di mashup tramite Facebook può rappresentare uno dei punti di forza del MashupEditor, all'interno di Facebook gli utenti con interessi affini (es. squadre di calcio, gruppi musicali, film, ecc...) condividono le proprie esperienze tramite l'iscrizione a *gruppi* o *pagine*. Questi strumenti consentono lo scambio di esperienze tra gli utenti, i quali contribuiscono tramite la pubblicazione di contenuti, l'invio dei commenti, l'utilizzo del *like* per la valutazione dei contenuti. La nascita di gruppi e/o pagine che riguardano i mashup, dove condividere le applicazioni create attraverso il MashupEditor può contribuire a far conoscere, e quindi utilizzare, la nostra piattaforma a tutti gli utenti interessati a fare delle esperienze con il mondo dei mashup di applicazioni Web.

6.1 Facebook

Facebook è un servizio di rete sociale lanciato nel febbraio 2004, gestito e di proprietà di *Facebook, Inc.* Il sito, fondato a Cambridge negli Stati Uniti da Mark Zuckerberg e dai suoi compagni di college Eduardo Saverin, Dustin Moskovitz e Chris Hughes, era originariamente stato progettato esclusivamente per gli studenti dell'Università di Harvard. Ben presto fu reso accessibile anche agli studenti di altre scuole nella zona di Boston, della Ivy League e della Stanford University.
Successivamente fu aperto anche agli studenti delle scuole superiori e poi a chiunque avesse più di 13 anni. Da allora Facebook raggiunse un enorme successo, è diventato il secondo sito più visitato al mondo preceduto solo da

Google. Facebook è disponibile in oltre 70 lingue e conta più di 850 milioni di utenti attivi che effettuano l'accesso almeno una volta al mese.
Per poter usufruire del sito, occorre prima registrarsi fornendo alcuni dati: nome, cognome, data di nascita e indirizzo email.
Una volta registratosi, l'utente può procedere a creare il proprio profilo personale. È possibile inoltre aggiungere altri utenti tra i propri amici, prendere parte a dei gruppi in base ai propri interessi, scambiarsi messaggi di posta o tramite una chat, condividere immagini, video ed altri contenuti ed utilizzare varie applicazioni presenti sul sito.
Per personalizzare il proprio profilo l'utente può caricare una foto, chiamata immagine del profilo, con la quale può rendersi riconoscibile. Può inoltre fornire ulteriori informazioni, come la città di nascita e quella di residenza, la scuola frequentata o il proprio datore di lavoro, l'orientamento religioso e quello politico, la propria situazione sentimentale e molte altre.

6.2 Facebook Platform

La Facebook Platform mette a disposizione degli sviluppatori una serie di API e di tool, che permettono di usufruire delle funzionalità di Facebook. La Facebook Platform è stata lanciata il 24 maggio 2007, all'inizio forniva il supporto per la creazione di applicazioni che funzionavano solo all'interno della piattaforma, poi nel tempo sono stati introdotti una serie di strumenti per poter usufruire delle funzionalità proprie del social network anche all'interno di siti Web esterni, oppure all'interno di device mobili come smartphone, telefonini o tablet.
Ecco alcune statistiche sulla piattaforma (Maggio 2010):

- Più di un milione di sviluppatori e imprenditori provenienti da oltre 180 paesi;
- Più di 550.000 applicazioni attualmente disponibili sulla Facebook Platform;

- Ogni mese, oltre il 70% degli utenti di Facebook utilizza le applicazioni;
- Più di 250.000 siti web sono integrati con Facebook;
- Più di 100 milioni di utenti Facebook al mese interagisce con Facebook da siti esterni.

Sono disponibili attualmente diversi modi per interagire con la Facebook Platform:

- *Graph API*: è il nucleo della piattaforma Facebook, permette agli sviluppatori di leggere e scrivere dati all'interno di Facebook. *Graph API* consente di accedere in maniera semplice al *Facebook Social Graph*, ossia alle entità che rappresentano gli oggetti presenti in Facebook (ad esempio persone, foto, eventi o pagine) e le connessioni tra di loro (ad esempio le relazioni di amicizia, i contenuti condivisi o i tag nelle foto);
- *Authentication*: la piattaforma fornisce un meccanismo di autenticazione che consente alle applicazioni sviluppate all'esterno di interagire con le *Graph API* per conto degli utenti di Facebook, inoltre fornisce un meccanismo di *single-sign* con la piattaforma che può essere utilizzato nei siti Web oppure nelle applicazioni desktop o mobile;
- *Social Plugins*: sono dei plugin che si possono inserire all'interno dei siti Web, che consentono agli sviluppatori di fornire agli utenti delle funzionalità presenti in Facebook anche all'interno dei siti Web esterni, semplicemente includendo poche righe di codice HTML. Tutti i plugin sono delle estensioni di Facebook e sono progettati in maniera tale che nessun dato che riguarda l'utente sia condiviso con i siti dove vengono utilizzati. Sono disponibili diversi *Social Plugins* ricordiamo *Like Button, Recommendations, Comments* ed *Activity Feed*;
- *Open Graph Protocol*: grazie a questo protocollo gli sviluppatori possono integrare le loro pagine Web all'interno del Facebook Social Graph, ciò consente a tali pagine di poter far utilizzare le funzionalità degli

oggetti nativi di Facebook e quindi possono essere ricercate, condivise ed utilizzate all'interno della piattaforma. *Open Graph* infatti può funzionare come un Search Engine, le pagine devono essere modificate aggiungendo i metadata che sono necessari per la ricerca.

Per poter utilizzare le funzionalità di Facebook su un sito Web esterno è necessario utilizzare un SDK JavaScript, che si installa all'interno dell'applicazione Web e si occupa di interagire con la piattaforma Facebook, facendo da connettore per tutti i tipi di funzionalità che sono utilizzate.
Il codice necessario è il seguente:

```
<div id="fb-root"></div>
<script>(function(d, s, id) {
    var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) return;
   js = d.createElement(s); js.id = id;
    js.src = "//connect.facebook.net/it_IT/all.js#xfbml=1&appId=
        APP_ID";
    fjs.parentNode.insertBefore(js, fjs);
 }(document, 'script', 'facebook-jssdk'));
</script>
```

Facebook consente di utilizzare le sue funzionalità con diversi linguaggi, infatti è possibile inserire le funzionalità aggiungendo del codice HTML5 alle pagine Web, oppure di utilizzare un linguaggio di markup chiamato FBML[1] (*Facebook Markup Language*), per impiegare il FBML bisogna inserire il namespace di riferimento all'interno del tag $< HTML >$ della pagina Web che si sta creando:

```
<html xmlns:fb="http://ogp.me/ns/fb#">
```

Ad esempio, per utilizzare il Plugin *Send Button* che consente di inviare ad un amico un messaggio contente il collegamento al sito riferito dal plugin, basta inserire il seguente codice all'interno della pagina Web:

- HTML5:

[1]Facebook Markup Language

```
<div class="fb-send" data-href="http://example.com"></div>
```

- FBML:

```
<fb:send href="http://example.com"></fb:send>
```

L'utilizzo dell'SDK Javascript consente di utilizzare e definire una serie di handler per gli eventi che vengono lanciati dai vari social plugin, è possibile definire delle callback personalizzate che vengono chiamate automaticamente quando viene lanciato un determinato evento dal plugin.
Il plugin *Send Button* lancia un evento quando viene inviato il messaggio, per catturarlo basta infatti inserire ed eseguire il seguente codice Javascript nella pagina:

```
FB.Event.subscribe('message.send',
    function(response) {
        alert('You sent the URL: ' + response);
    }
);
```

Per eliminare un event handler basta invocare la funzione *FB.Event.unsubscribe*, che eliminerà la corrispondenza tra l'evento lanciato dal plugin e la funzione callback personalizzata che è stata definita per quell'evento.
Attraverso il sito degli sviluppatori è possibile accedere alle funzionalità di sviluppo. Sono presenti inoltre un *debugger* online ed uno strumento visuale il *Graph API Explorer*, che permette di utilizzare direttamente il *Facebook Social Graph* per testare le operazioni che si vogliono integrare.

6.3 MashupEditor Application

Il primo passo che è stato compiuto per integrare il MashupEditor con Facebook, è stato quello di creare un applicazione Facebook che rappresenta l'interfaccia tra le due piattaforme. Per creare una nuova applicazione è stato sufficiente collegarsi al sito degli sviluppatori ed inserire alcune informazioni sull'applicazione, che è stata chiamata "*MashupEditor*" (figura 6.1).

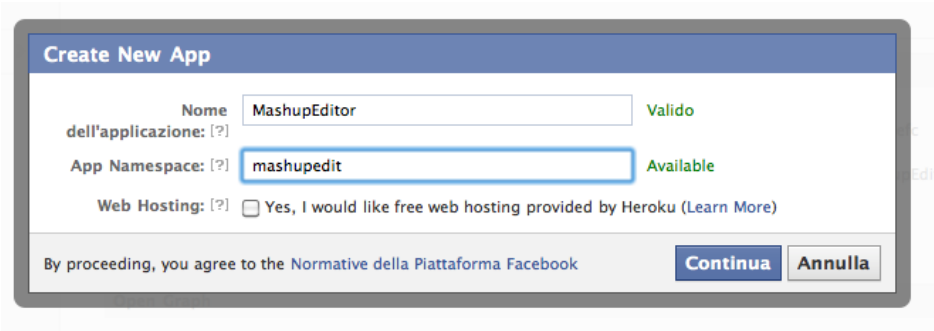

Figura 6.1: Schermata per la creazione dell'applicazione MashupEditor

Una volta creata l'applicazione vengono forniti un *ID applicazione* (*APP_ID*) ed un *ID applicazione segreto*, che devono essere inseriti nei parametri di configurazione dell'SDK Javascript.
L'applicazione è configurabile tramite un pannello di controllo, dal quale si può accedere a diverse funzioni (figura 6.2). La piattaforma inoltre mette a disposizione un utente fittizio chiamato "*test user*" per la fase di testing della propria applicazione.

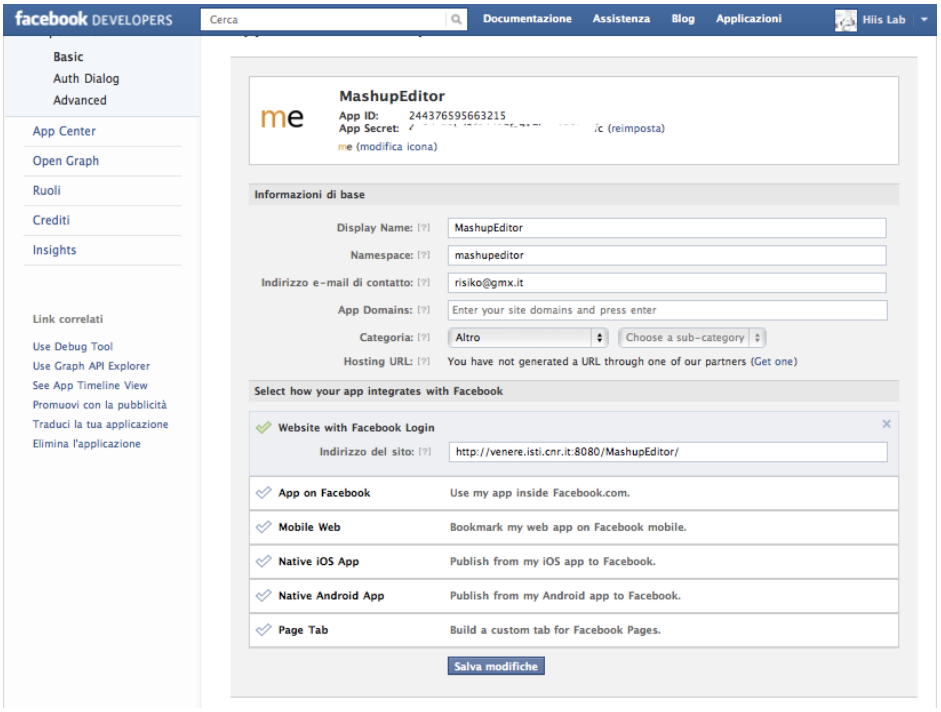

Figura 6.2: Pannello di controllo dell'applicazione MashupEditor

6.4 Login tramite Facebook

Per utilizzare la funzionalità di login bisogna indicare nel pannello di controllo dell'applicazione l'URL base del sito Web che si intende collegare a Facebook. Nel MashupEditor una volta effettuato il login tramite il Login Button, l'utente può lavorare all'interno dell'ambiente grafico come se avesse fatto il login nella form originale della piattaforma di mashup.
Il codice della home page del sito è stato modificato inserendo il *Login Button* tramite l'utilizzo del FBML:

```
        <fb:login-button autologoutlink="true"
                         perms="email,status_update,
                             publish_stream">
</fb:login-button>
```

L'attributo "*perms*" indica che quando l'utente utilizza per la prima volta il *Login Button* all'interno del MashupEditor, dovrà dichiarare se vuole concedere all'applicazione di leggere alcuni dei suoi dati personali, in questo caso l'email, se vuole permettere all'applicazione di poter aggiornare il suo stato su Facebook, e se l'applicazione può pubblicare delle notizie sulla sua bacheca personale.
É possibile aggiungere altri attributi tra cui "*show-faces*" che consente di mostrare le immagini degli utenti che utilizzano l'applicazione, e "*registration-url*" dove si può indicare la URL da mostrare ad un utente che non è registrato al sito Web.
Sono stati definiti i seguenti event handler:

```
FB.Event.subscribe('auth.login', function(response) {
      fb_login(response);
 });

  function fb_login(response){
       if(response.status == 'connected'){

        var uid = response.authResponse.userID;
             var accessToken = response.authResponse.accessToken
                 ;
           var data = {
```

```
                //'cmd' : 'load_me',
                  'uid': uid ,
                  'access_token': accessToken,
                  'fb_op': 'login'          };
          command('load_me', data, callb);

    } else if (response.status === 'not_authorized') {
        // utente loggato ma non ha dato l'autorizzazione all'
            applicazione
        return;
    } else {
        //utente nn loggato in fb
        return;
    }
}

FB.Event.subscribe('auth.logout', function (response){
      fb_logout(response);
});

FB.getLoginStatus(function(response) {
      fb_login(response);
});
```

Il primo handler viene chiamato dopo che l'utente fa il login tramite il Login Button (figure 6.3 e 6.4). Il plugin effettua il login su Facebook, se quest'ultimo è andato a buon fine, la funzione di callback avrà "response.status = "*connected*" ", viene così visualizzata all'utente la schermata principale dell'ambiente grafico (figura 6.5).

Se l'utente si è loggato con successo su Facebook la piattaforma invierà nella response un *Token*, che viene utilizzato nei successivi accessi alle funzionalità di Facebook per controllare se l'utente è autorizzato a svolgere determinate operazioni e la validità della sessione dopo il login.

Il secondo handler controlla la funzionalità di logout e viene richiamato quando l'utente fa il logout dalla piattaforma attraverso il bottone di logout di Facebook presente all'interno dell'ambiente grafico.

Il terzo handler invece controlla automaticamente se l'utente è già loggato su Facebook, al momento del caricamento della home page del MashupEditor ed in caso positivo, lancia la procedura di login al MashupEditor.

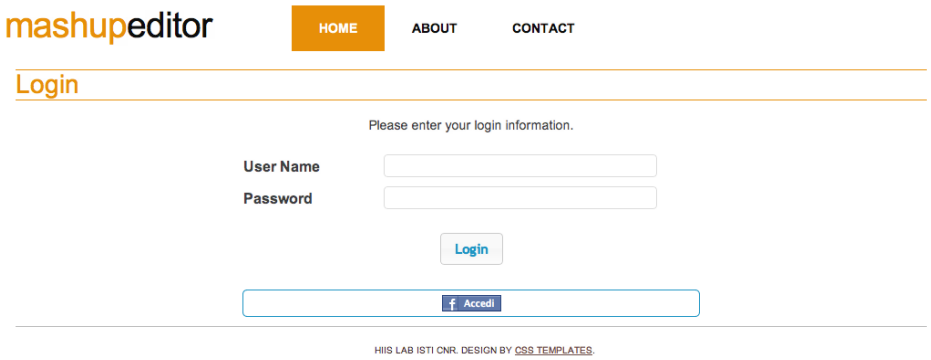

Figura 6.3: Home Page del MashupEditor che mostra il Login Button

Figura 6.4: Finestra dove si inseriscono le credenziali per l'autenticazione in Facebook

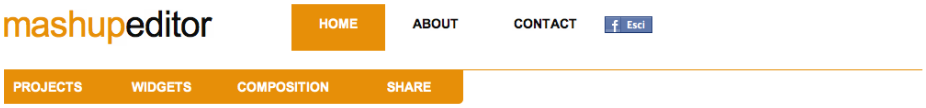

Figura 6.5: Menù principale dell'applicazione che visualizza il tasto Logout da Facebook

6.5 Share e Like Button

Il secondo plugin inserito è stato quello del *Like Button*, questo plugin consente all'utente di esprimere un voto positivo per un determinato oggetto Facebook e, contemporaneamente, pubblica tale voto sulla bacheca dell'utente. All'interno del MashupEditor è stato inserito come strumento di votazione dei mashup.

É ormai prassi all'interno dei siti Web che integrano tale plugin, valutare il livello di gradimento di un determinato contenuto in base al numero di *like* ricevuti. Il *Like Button* è stato pensato per sostituire il più anziano *Share Button*, quest'ultimo è stato il primo plugin che ha consentito agli utenti di un sito Web esterno di poter condividere nella propria bacheca di Facebook il contenuto desiderato (figura 6.8).

Questi due plugin appaiono solo quando vi è un mashup aperto all'interno dell'editor, in quanto ogni mashup ha un ID e quindi una URL univoca all'interno del MashupEditor. La URL rappresenta il fattore utilizzato dal plugin per distinguere i contenuti della piattaforma. Lasciare il plugin a disposizione degli utenti quando non ci sono progetti aperti può causare votazioni e condivisioni dell'URL della home page, ciò significherebbe raccogliere e memorizzare dei voti che non hanno significato (figura 6.6).

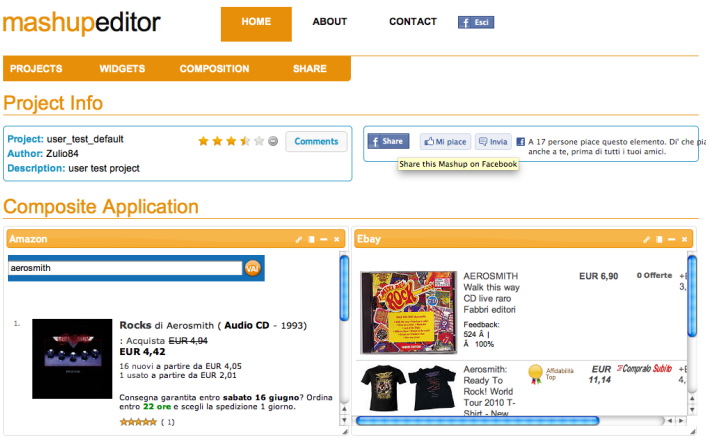

Figura 6.6: Schermata dell'editor con un mashup aperto

Il tasto *Share* rimane più conosciuto ed utilizzato rispetto al *Like Button*, che dovrebbe sostituirlo, perciò è stato deciso di integrarli entrambi all'interno dell'ambiente grafico. Il *Like Button* è stato inserito insieme al *Send Button*, che consente all'utente di inviare attraverso un messaggio, il link relativo al mashup corrente ai suoi amici su Facebook.
Per inserire il tasto *Share* non esiste un comando in FBML, perciò è stato inserito all'interno del codice HTML dell'ambiente grafico tramite un elemento di tipo anchor, definendo poi lo stile tramite il CSS:

```
<a id="fb_shareMashup" class="fb_shareMashup" title="Share this
    Mashup on Facebook"
      onclick='postProjectOnFB(); return false;'></a>
```

Quando l'utente clicca sul tasto si richiama la funzione JavaScript messa a disposizione dall'SDK:

```
function postProjectOnFB(){
    var obj = {
        method: 'stream.publish',
        message: 'you\'re sharing a mashup: ' + fb_pn ,
        link: domain +'/MashupEditor/CommandServlet?p_id='+p_id+
            '',
        picture: domain + '/images/logo1p_bg.png',
        name: 'Mashup: ' + fb_pn,
        caption: 'Author: ' + fb_a,
        description: 'Description: ' + fb_d,
        display: 'popup'
    };

    function callbackShare(response) {
        if(response){
  // debug
      \$( "#outputDiv").html ("Post ID: " + response['post_id'
          ]);
        }
    }
    FB.ui(obj, callbackShare);
}
```

La funzione FB.ui crea e mostra automaticamente la finestra di condivisione del link su Facebook (figura 6.7).

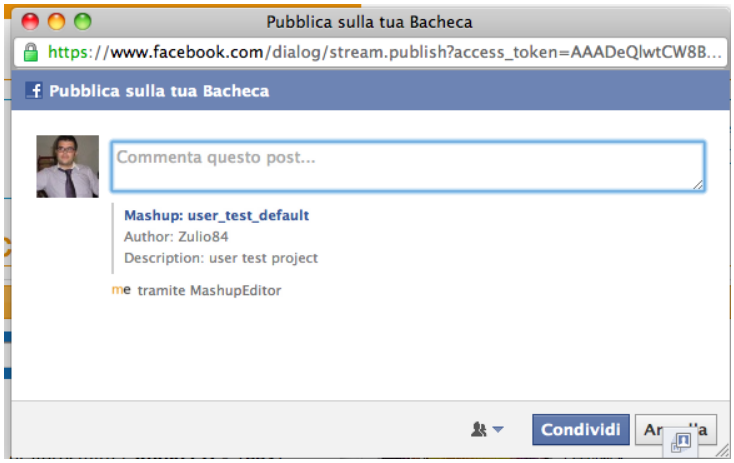

Figura 6.7: Schermata per la condivisione del mashup su Facebook

Figura 6.8: Box del mashup condiviso all'interno della bacheca di Facebook

Per utilizzare il *Like Button* è stato inserito all'interno della pagina HTML dell'editor il seguente codice FBML:

```
<fb:like href="current Mashup URL"
    send="true" width="450" show_faces="false"></fb:like>
```

Il parametro "*send*" con valore uguale a *true* significa che verrà visualizzato anche il *Send Button*. Si possono configurare anche altri parametri tra i quali il tipo di visualizzazione, il tipo di label da mostrare oppure il parametro "*show-faces*" come nel Login Button.
L'SDK consente di definire degli event handler quando si utilizza il *Like Button*:

```
FB.Event.subscribe('edge.create',
                   function(response) {
                       fb_add_like(response);
                   });
 FB.Event.subscribe('edge.remove',
                   function(response) {
                       fb_remove_like(response);
                   });
```

Il primo evento viene sollevato quando l'utente esprime la sua preferenza per il mashup attraverso il plugin, la preferenza viene conteggiata all'interno del database del Mashup Support. Il secondo evento accade quando l'utente elimina tale preferenza, che viene quindi eliminata dal database.

Le seguenti figure mostrano l'utilizzo del *Like Button* e del *Send Button*.

Figura 6.9: Esprimo la preferenza tramite il Like Button

Figura 6.10: Box generato dall'azione like all'interno della bacheca di Facebook

Figura 6.11: Utilizzo del Send Button

Figura 6.12: L'utente destinatario del messaggio riceve il link al mashup

6.6 Comments

Il plugin *Comments* consente agli utenti di inserire dei commenti all'interno di un sito Web, offrendo agli amministratori anche gli strumenti per la moderazione dei commenti, e la condivisione di tali commenti sul sito di Facebook. La gestione dei commenti avviene nella stessa maniera con cui la piattaforma Facebook tratta i post degli utenti, i commenti che vengono ritenuti più interessanti oppure quelli degli amici, amici di amici oppure quelli che hanno un numero elevato di *like* o di commenti, sono mostrati in evidenza all'interno del plugin, quelli meno interessanti o segnalati come spam vengono nascosti. È comunque possibile visualizzare i commenti in ordine cronologico. Quando l'utente inserisce un commento può scegliere se inviarlo alla piattaforma Facebook, e in caso affermativo il commento apparirà tra le notizie dei suoi amici, insieme al link alla URL del sito Web esterno.
Le persone possono commentare i commenti inseriti da altri utenti direttamente all'interno del plugin, oppure anche all'interno di Facebook. Il plugin provvederà automaticamente all'aggiornamento dei contenuti.
Il plugin è stato inserito nella piattaforma di Mashup utilizzando il seguente codice FBML:

```
<fb:comments href="Current Mashup URL"
                                    num_posts="2" width="470">
</fb:comments>
```

Il parametro "*num_posts*" indica il numero di commenti da visualizzare di default. Gli event handler che sono stati associati a questo plugin sono i seguenti:

```
FB.Event.subscribe('comment.create',
                   function(response) {
                       fb_add_comment(response);
                   });
 FB.Event.subscribe('comment.remove',
                   function(response) {
                       fb_remove_comment(response);
                   });
```

Il primo evento viene lanciato quando un utente inserisce un commento all'interno del plugin, in questo caso la funzione di callback *fb_add_comment* provvede a memorizzare il commento anche all'interno del database della piattaforma di Mashup. Il secondo evento viene lanciato quando un utente cancella un proprio commento, la funzione di callback *fb_remove_comment* provvede a cancellare il commento dal database.
Nelle seguenti figure viene mostrato l'utilizzo del plugin con la piattaforma.

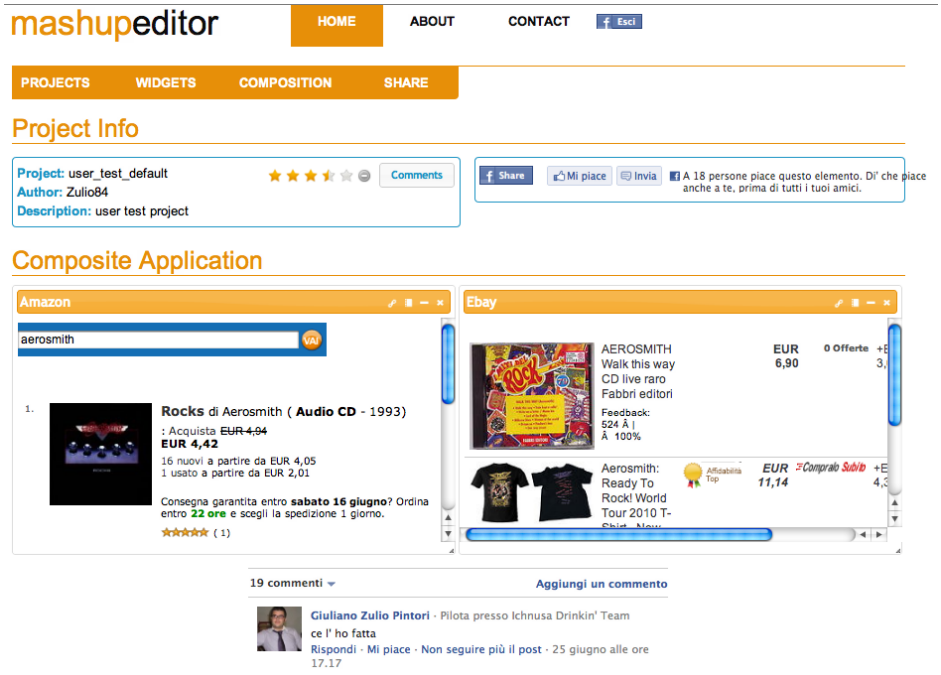

Figura 6.13: Schermata dell'editor che contiene il plugin Comments

Figura 6.14: Inserimento di un nuovo commento

Figura 6.15: Il commento pubblicato appare su Facebook

Figura 6.16: Inserimento di un commento ad un commento

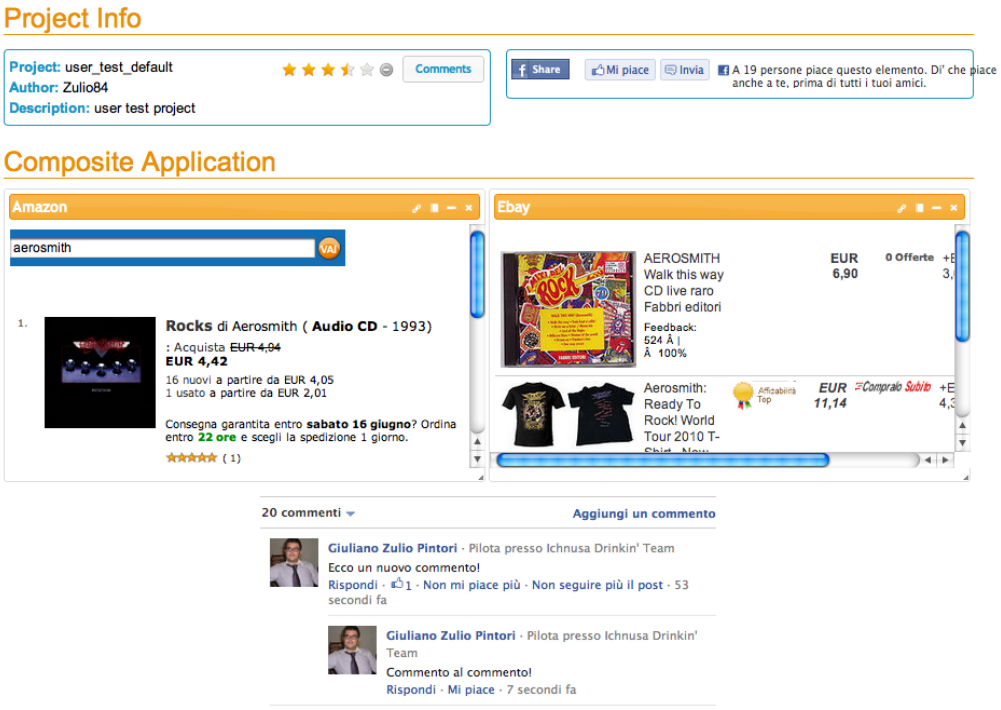

Figura 6.17: Il plugin aggiorna il suo contenuto con il nuovo commento inserito su Facebook

6.7 Open Graph Protocol

I plugin *Like Button* e *Comments* sono stati impiegati per dare la possibilità all'utente di condividere le proprie esperienze, soluzioni e commenti quando utilizza dei mashup che creati in precedenza.
Per condividere su Facebook le notizie riguardanti i nuovi mashup creati e condivisi sul repository della piattaforma, è stato scelto di utilizzare il protocollo *Open Graph* per pubblicare tali notizie sul social network.
Il protocollo consente di definire un insieme di *Actions* ed *Objects*, che rappresentano le entità di base del protocollo, le *Actions* rappresentano le iterazioni che gli utenti possono fare attraverso l'applicazione, mentre gli *Objects* sono le entità che su cui si possono effettuare le *Actions*. Per utilizzarli bisogna definirli all'interno dell'*APP Dashboard*.
Ad esempio un applicazione che riguarda la cucina potrebbe avere l'azione "*Cook*" e gli oggetti "*Recipe*" oppure "*Menù*".
Per l'applicazione *MashupEditor* è stato definito l'oggetto "*Mashup*" (figura 6.21), che rappresenta il progetto creato, mentre l'azione che compie un utente è stata chiamata "*Create*" (figure 6.18 e 6.19).

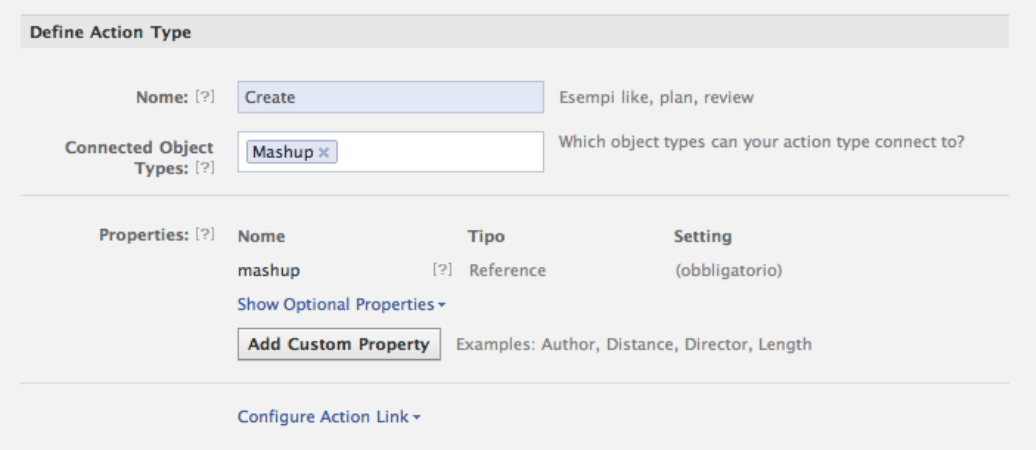

Figura 6.18: Definizione dell'Action "Create" 1

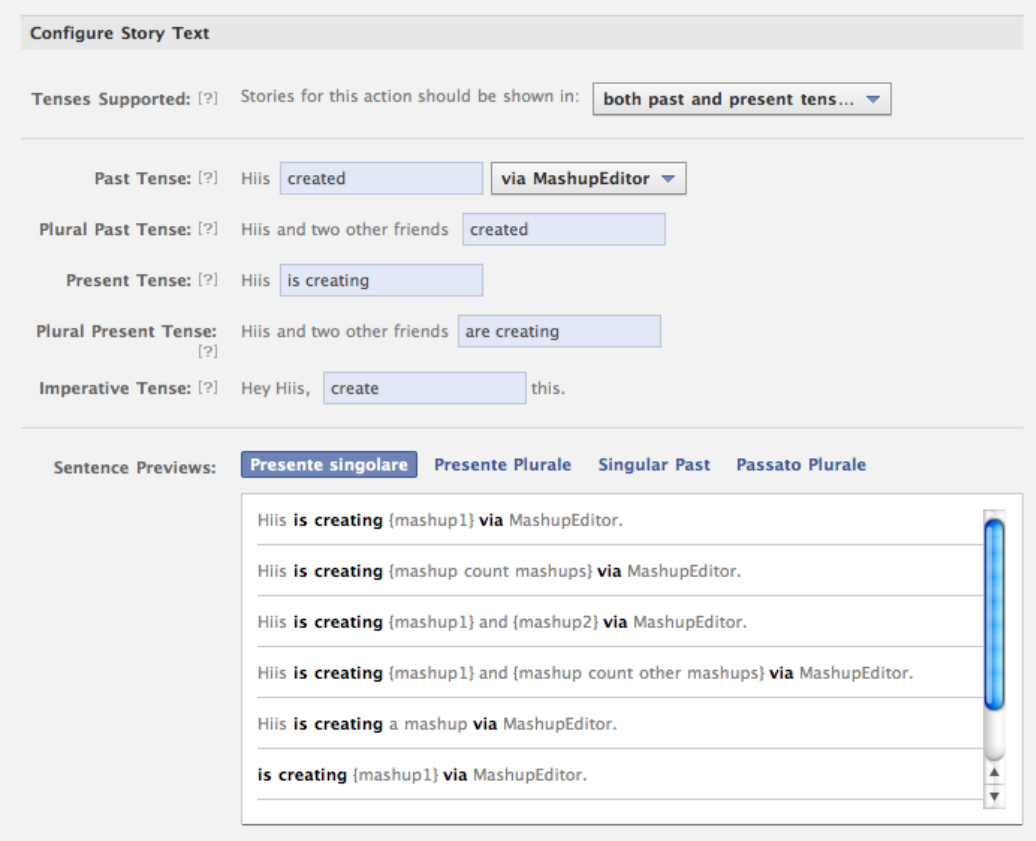

Figura 6.19: Definizione dell'Action "Create" 2

L'APP Dashboard consente di definire l'azione inserendo alcune informazioni riguardanti il tipo di oggetti utilizzati e, di definire la struttura semantica delle notizie che verranno pubblicate sul social network. Inoltre si può personalizzare l'aspetto delle notizie all'interno della time-line dell'utente, è possibile inserire l'elenco delle *Actions* effettuate come lista, tabella, all'interno di una mappa o semplicemente visualizzare il contatore.

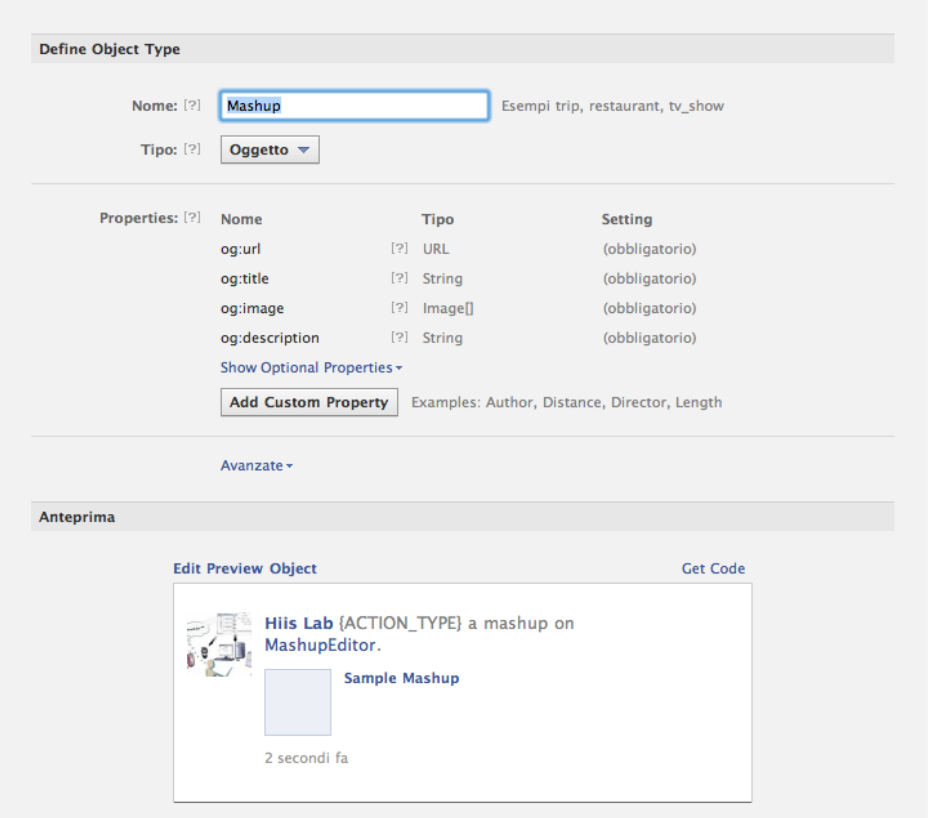

Figura 6.20: Definizione dell' Object "Mashup"

I campi che sono definiti all'interno delle *Properties* devono essere popolati all'interno della pagina Web inserendoli all'interno di tag HTML $< meta >$, le informazioni inserite nei tag vengono poi utilizzate per completare la struttura delle notizie che appariranno sul social network.

Esempio dei tag $< meta >$ che vengono inseriti nella pagina HTML:

```
 <meta property="fb:app_id" content="244376595663215" />
<meta property="og:type" content="mashupeditor:mashup" />
<meta property="og:title" content="user_test_default" />
 <meta property="og:image"
content="http://venere.isti.cnr.it:8080/MashupEditor/images/
    logo1p_bg_200.png" />
<meta property="og:description" content="user test project" />
<meta property="og:url"
 content="http://venere.isti.cnr.it:8080/MashupEditor/
     CommandServlet?p_id=1339686564158">
```

Una volta definiti *Actions* e *Objects* si inseriscono i tag all'interno della pagina Web. Quando l'utente esegue l'azione definita all'interno dell'applicazione, l'applicazione stessa invoca le *Graph API* che pubblicano una nuova istanza dell'azione compiuta collegando l'utente con l'oggetto utilizzato. La sequenza di azioni che vengono eseguite viene elencata nella figura **??**.

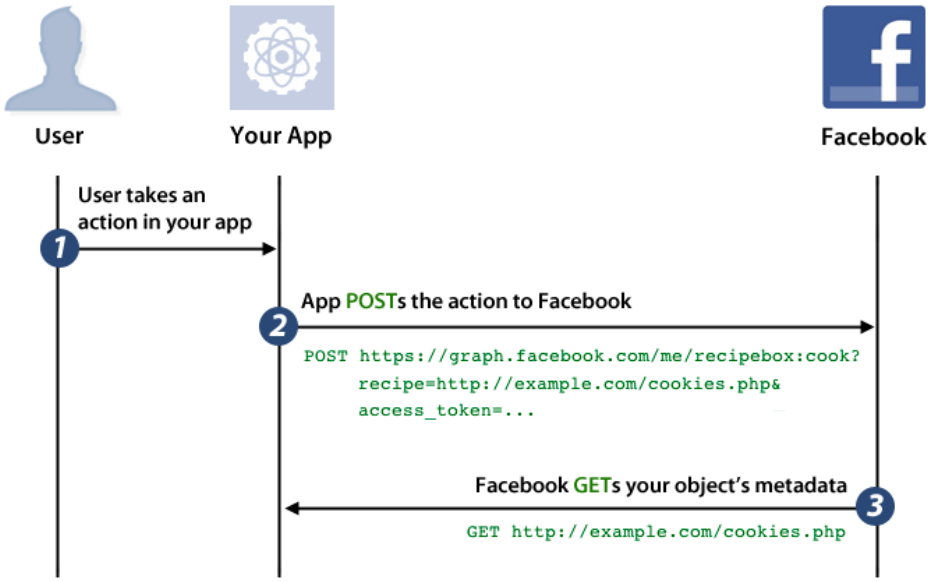

Figura 6.21: Schema iterazioni con il protocollo Open Graph

All'interno del MashupEditor questo meccanismo è stato implementato nel seguente modo:

1. L'utente che ha creato il progetto lo condivide all'interno del repository, indicando la volontà di condividere la notizia su Facebook:

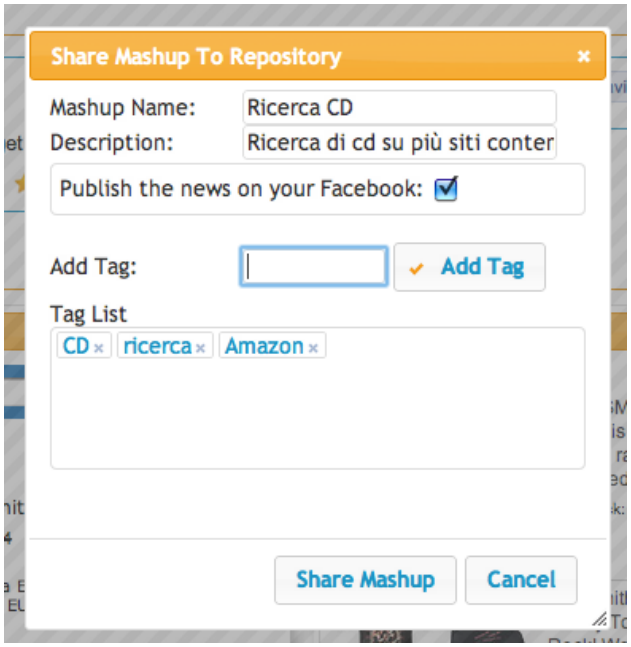

Figura 6.22: Finestra inserimento informazioni sul mashup da condividere

2. La piattaforma di Mashup effettua la POST dell'azione verso Facebook attraverso una chiamata HTTP:

```
POST https://graph.facebook.com/me/mashupeditor:create ?
    mashup=OBJECT_URL
&access_token=ACCESS_TOKEN
```

   dove *OBJECT_URL* è la url del mashup all'interno della piattaforma, mentre *ACCESS_TOKEN* è il token che l'utente ha ricevuto quando ha fatto il login tramite Facebook;

3. La piattaforma Facebook effettua una chiamata HTTP GET alla *OBJECT_URL* da dove legge i *metadati* che sono inseriti all'interno della pagina Web, creando la connessione tra l'utente Facebook e l'oggetto *Mashup* tramite l'azione *Create*.

La notizia appena generata viene visualizzata in tutti i *Social Channel* del sito di Facebook:

- News Feed e Ticker degli amici:

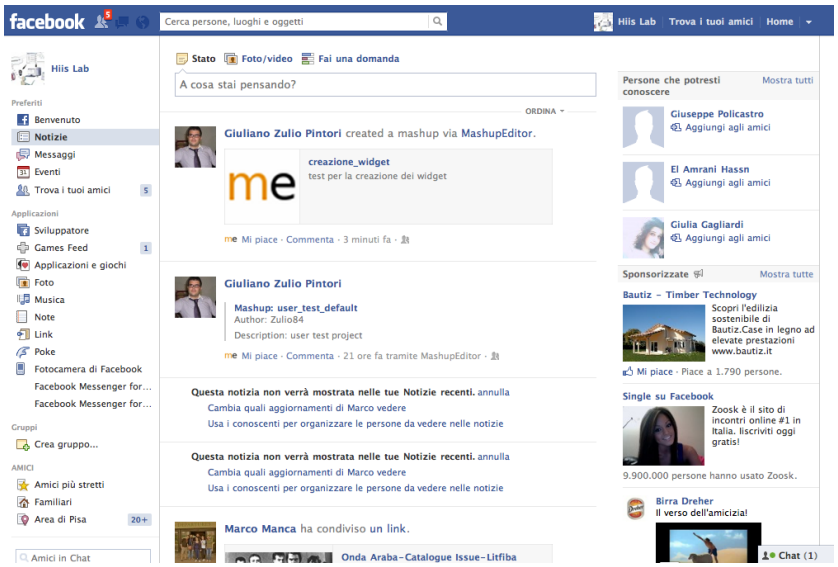

Figura 6.23: Notizia della creazione di un mashup sulla bacheca di un amico

- Timeline dell'utente:

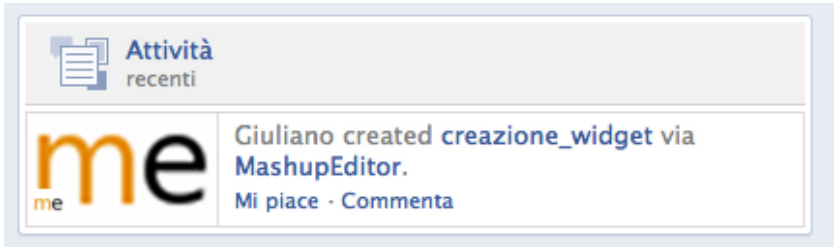

Figura 6.24: Notizia della creazione di un mashup sulla timeline dell'utente

- Timeline View dell'utente (figura 6.25), le notizie vengono inoltre aggregate per periodo, mese e anno (figure 6.27 e 6.26):

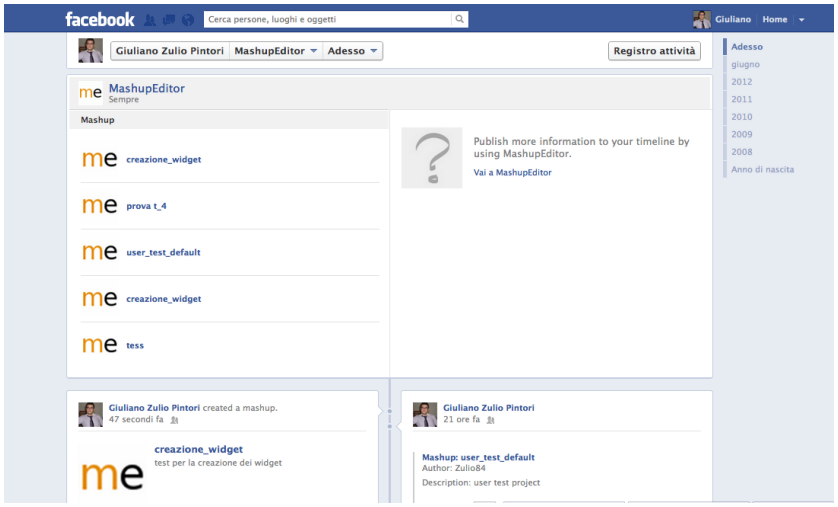

Figura 6.25: Elenco dei mashup creati dall'utente

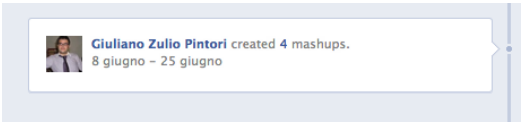

Figura 6.26: Riassunto dei mashup creati dall'utente per periodo temporale

Figura 6.27: Elenco dei mashup creati dall'utente durante il mese di giugno

Tutti i link ai mashup che vengono pubblicati su Facebook sono validi ed utilizzabili dagli utenti. Per rendere possibile ciò, la piattaforma è stata modificata affinché carichi automaticamente un mashup condiviso, rendendo possibile l'esecuzione all'utente che apre il link.
Se l'utente ha già utilizzato l'applicazione Facebook MashupEditor ed è connesso al social network, l'ambiente grafico mostrerà il mashup pronto per l'esecuzione. Se invece ha già utilizzato l'applicazione, ma non è autenticato su Facebook, viene mostrata la sua finestra di login e, una volta inserite le credenziali, viene caricato il mashup all'interno dell'ambiente grafico.
Infine, se un utente utilizza l'applicazione per la prima volta, verrà visualizzata la finestra di login di Facebook. Dopo questo, dovrà rilasciare le autorizzazioni richieste dall'applicazione MashupEditor ed infine l'ambiente grafico caricherà il mashup richiesto.

Capitolo 7

Un esempio di applicazione

In questo capitolo viene presentato un esempio d'uso della nuova versione della piattaforma di mashup. L'esempio è articolato in due parti e mostra la cooperazione possibile tra due utenti della piattaforma nello sviluppo di un mashup.

7.1 Scenario

- *Parte 1: Creazione e condivisione di un mashup*:

  Giuliano vuole creare un nuovo mashup per la ricerca di cd musicali. Egli vuole utilizzare due diverse applicazioni Web e visualizzare i risultati della ricerca di entrambi i siti.
  La creazione dell'applicazione parte creando un nuovo widget del sito "*www.amazon.it*" utilizzando la creazione a partire dalla navigazione Web.
  Una volta creato il primo widget, decide di cercare all'interno del repository se è presente un widget del sito "*www.ebay.it*".
  La ricerca ha esito positivo così Giuliano può passare alla fase di connessione dei due widget.
  Una volta effettuata la connessione, Giuliano esegue il suo mashup.

Infine Giuliano effettua la condivisione del suo progetto all'interno del repository e su Facebook.

- *Parte 2: Caricamento e modifica di un mashup*:

  Alessandra vede la notizia relativa al mashup creato da Giuliano all'interno della sua bacheca di Facebook.
  Carica il mashup attraverso il link che trova nella notizia su Facebook.
  Decide che può arricchire il mashup aggiungendo un widget che ha salvato nella sua libreria personale e relativo al sito "*www.wikipedia.it*".
  Connette il suo widget agli altri due già presenti ed esegue il mashup, infine effettua la condivisione della nuova versione, informando Giuliano delle modifiche che ha effettuato.

7.2 Parte 1: Creazione e condivisione di un mashup

La creazione di una nuova applicazione mashup comincia definendo il nome e la descrizione della nuova applicazione (figura 7.1).

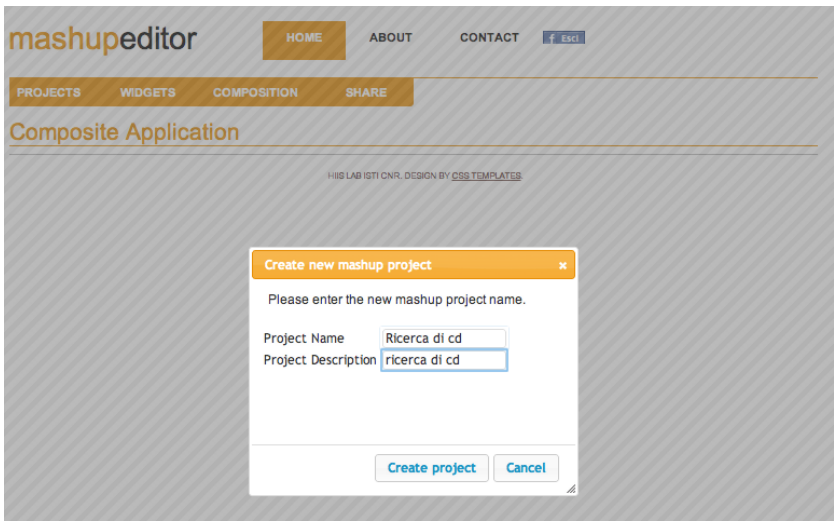

Figura 7.1: Creazione del progetto

Si procede poi alla creazione dei widget che compongono l'applicazione. In questo esempio il primo widget viene creato tramite la navigazione Web (figura 7.2). L'utente mentre naviga nel sito di *Amazon* effettua la selezione diretta delle componenti dell'applicazione Web che intende utilizzare.

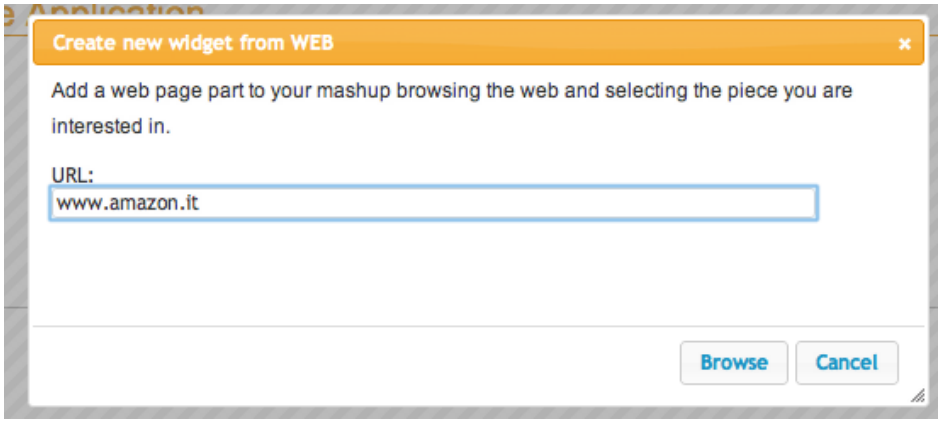

Figura 7.2: Creazione di un nuovo widget tramite la navigazione Web

Nel sito di *Amazon* viene eseguita una ricerca tramite la form (figura 7.3), quando l'utente fa la *submit*, gli script inseriti dal Proxy Server controllano quali elementi di input sono stati compilati dall'utente e provvede a inviarli automaticamente al MashupSupport per la creazione dell'*input component* del widget.

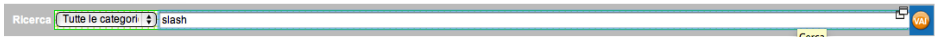

Figura 7.3: Selezione della form di ricerca sul sito di Amazon

All'interno della pagina dei risultati, l'utente seleziona i risultati a cui è interessato, e invia la sua selezione alla piattaforma di mashup tramite il comando che appare sul lato superiore destro degli elementi selezionati (figura 7.4).

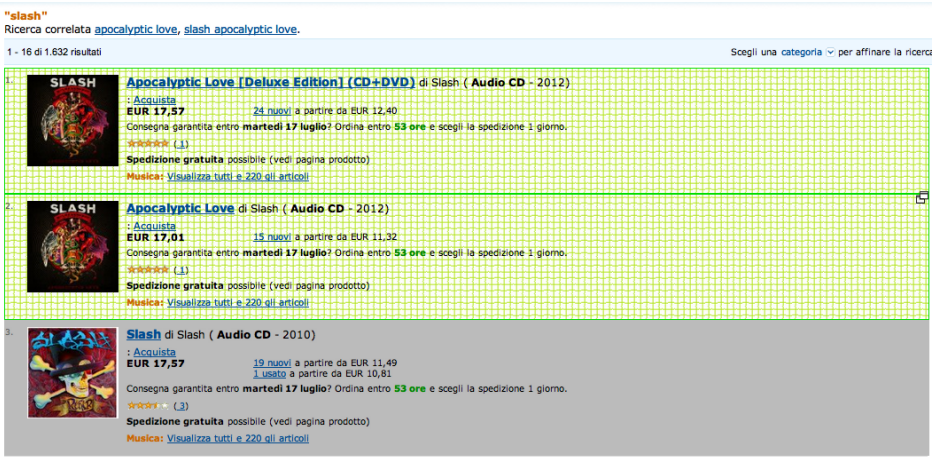

Figura 7.4: Selezione dei risultati della ricerca effettuata su Amazon

All'interno dell'ambiente grafico si inserisce il nome per il nuovo widget (figura 7.5).

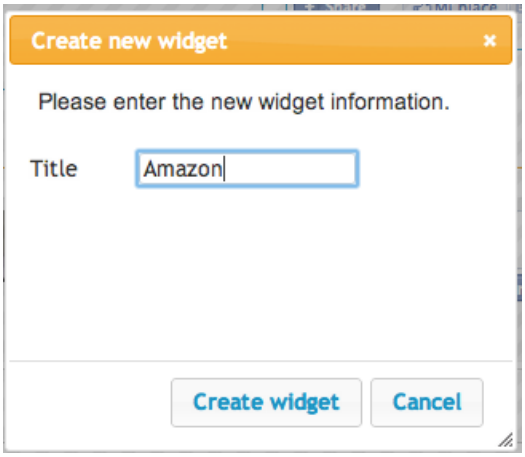

Figura 7.5: Inserimento del nome del widget

La figura 7.6 mostra l'aspetto dell'ambiente grafico dopo la creazione del widget.

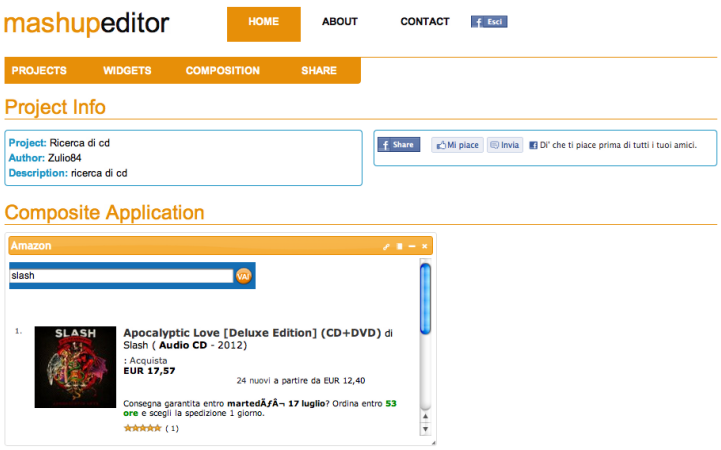

Figura 7.6: Il nuovo widget all'interno dell'ambiente grafico

Per la creazione del secondo widget l'utente utilizza il repository, effettua una ricerca di un widget per il sito di "*www.ebay.it*" e procede al caricamento all'interno del mashup (figure 7.7 e 7.8).

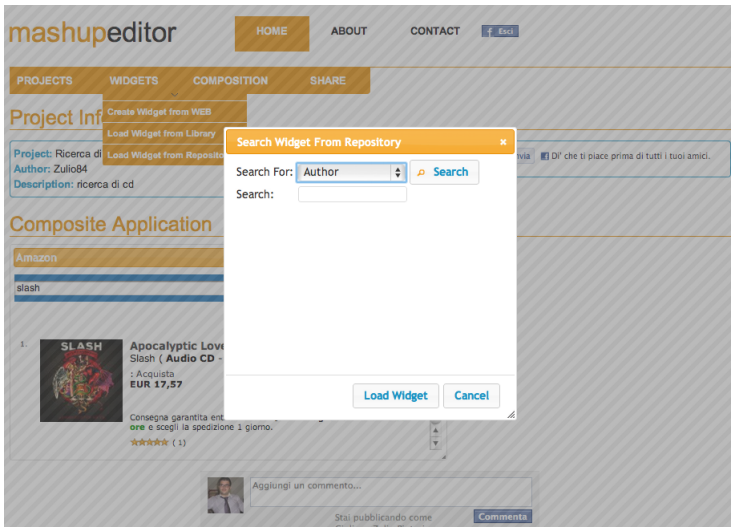

Figura 7.7: Finestra per la ricerca dei widget nel repository

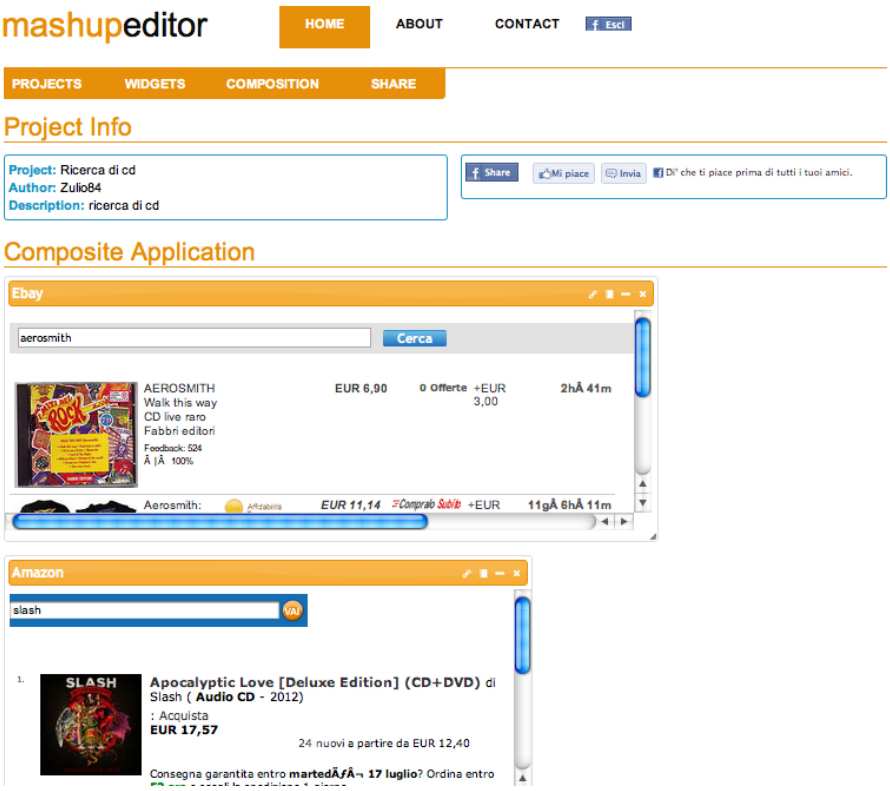

Figura 7.8: I due widget all'interno dell'ambiente grafico

Una volta inseriti i due widget all'interno del mashup, l'utente crea una connessione tra il widget *Amazon* e quello *Ebay*.
Per eseguire questa operazione l'utente attiva la funzione di registrazione delle sue azioni ed effettua un'operazione di *copia-incolla*: *copia* il contenuto del *text field* della search bar del widget di *Amazon* (figura 7.9) e lo *incolla* all'interno del *text field* della search bar del widget *Ebay* (figura 7.10). Una volta fatto questo ferma la registrazione (figura 7.11).
La piattaforma memorizza la connessione creata, e modifica l'aspetto del widget *Ebay*, nascondendone l'*input component* (figura 7.12).
Da questo momento aggiornando il contenuto della *text field* nella search bar del widget di *Amazon*, la piattaforma aggiorna il contenuto di entrambi i widget (figura 7.13).

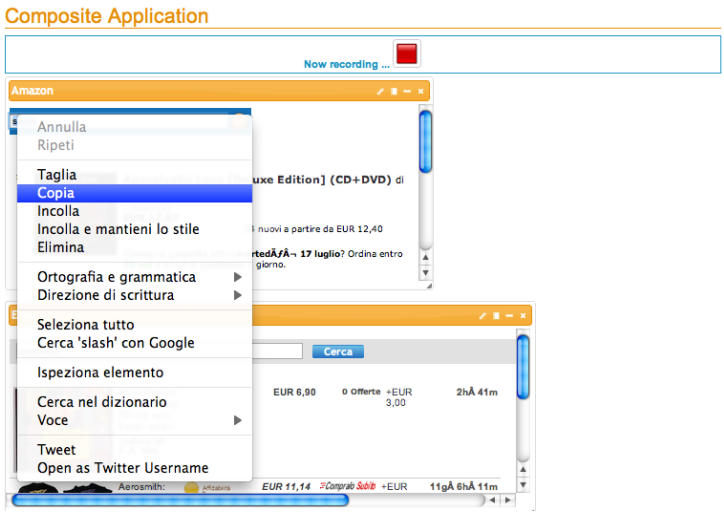

Figura 7.9: Operazione copia

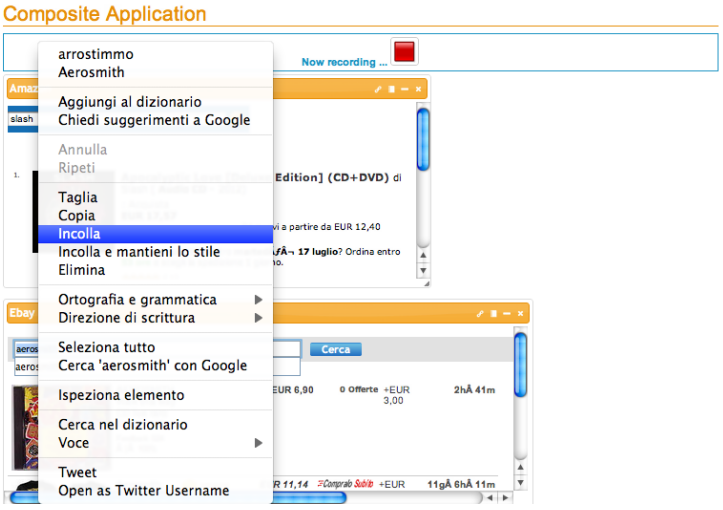

Figura 7.10: Operazione incolla

Figura 7.11: Comando di stop della registrazione

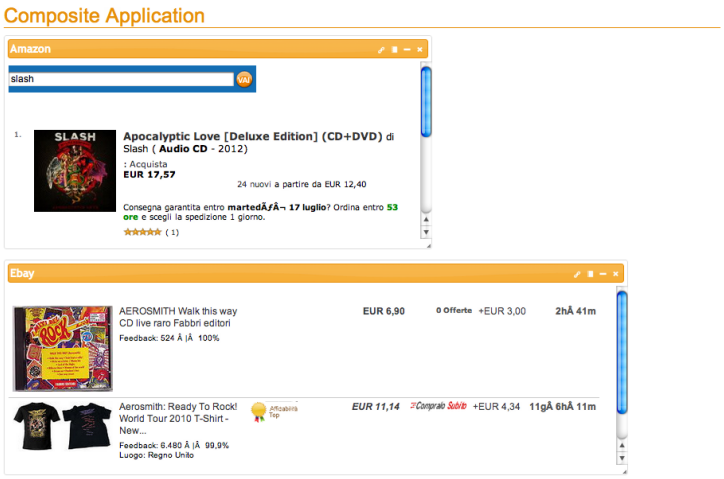

Figura 7.12: Aspetto dei due widget dopo la connessione

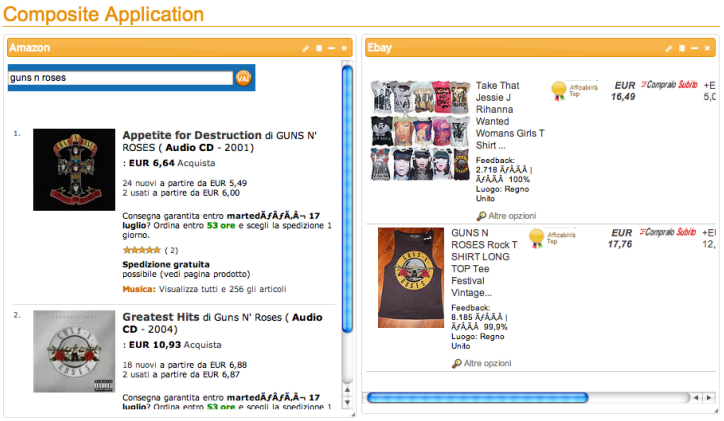

Figura 7.13: Risultato dell'esecuzione del mashup

L'utente condivide il mashup con gli utenti della piattaforma e con i suoi contatti su Facebook.
All'interno della finestra di condivisione inserisce le informazioni relative al mashup ed i tag per la ricerca all'interno del repository (figura 7.14).
Una volta che il mashup è disponibile agli altri utenti, l'ambiente grafico mostra all'utente i comandi per la votazione e il commento del mashup all'interno delle informazioni relative al progetto (figura 7.15).
Nella bacheca di Facebook dell'utente viene pubblicata automaticamente la notizia relativa alla creazione del mashup (figura 7.16).

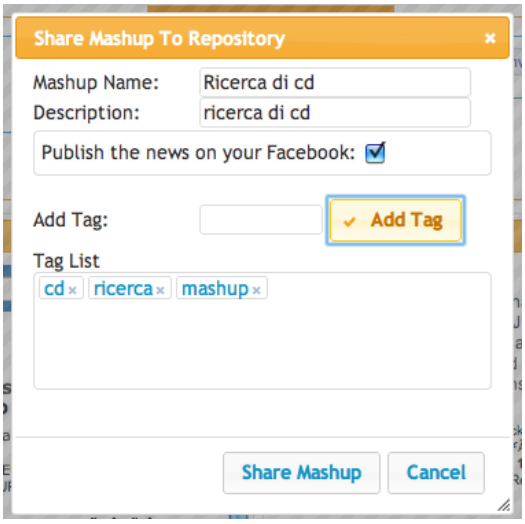

Figura 7.14: Finestra per l'inserimento delle informazioni del mashup da condividere

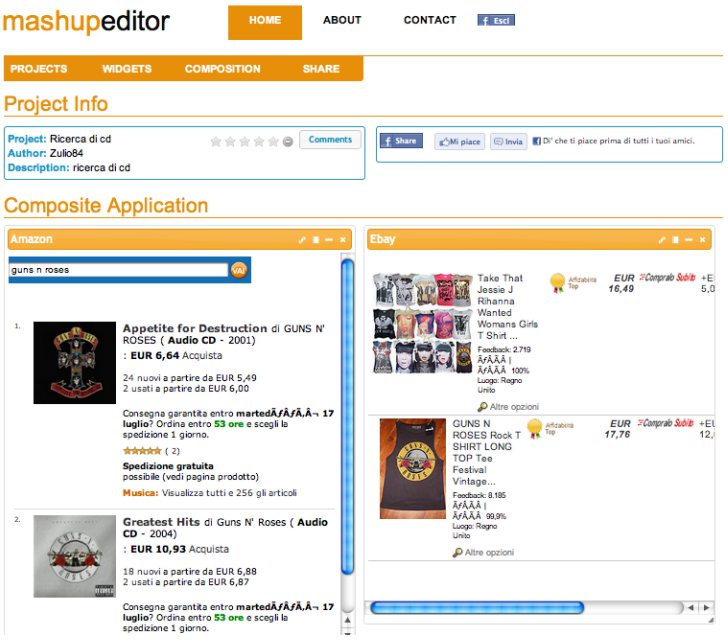

Figura 7.15: Le funzioni di voto e commento vengono visualizzate all'interno del'editor

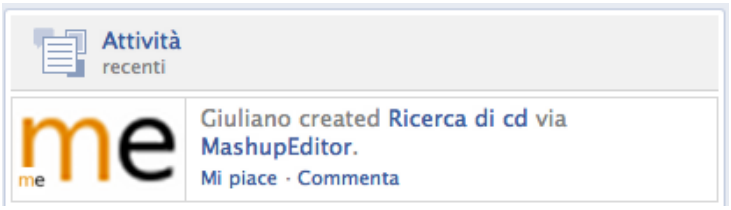

Figura 7.16: Notizia sulla creazione del mashup all'interno della bacheca dell'utente

7.3 Parte 2: Caricamento e modifica di mashup

Attraverso il link che ha trovato su Facebook, Alessandra carica il mashup condiviso da Giuliano (figure 7.17 e 7.18).

Figura 7.17: Link sulla bacheca di Facebook

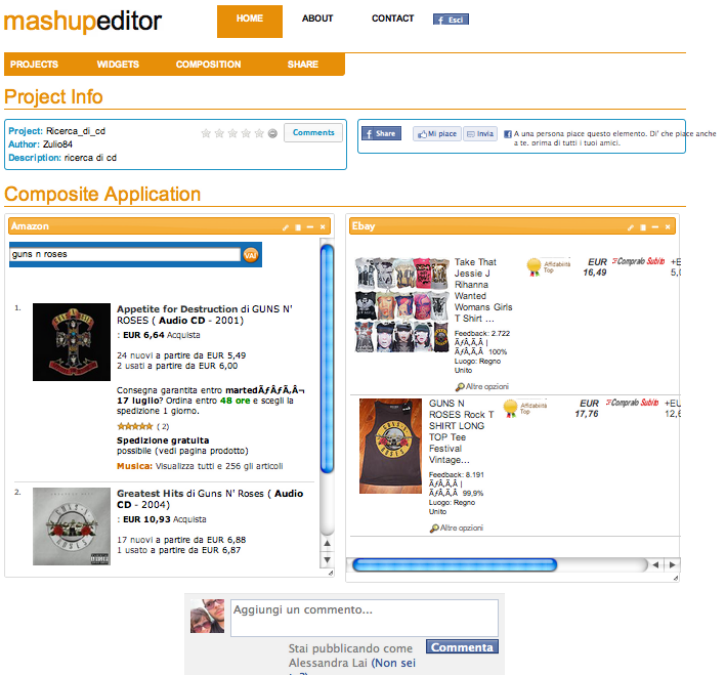

Figura 7.18: L'ambiente grafico mostra il mashup caricato

Alessandra carica il widget di "*Wikipedia*" dalla sua libreria personale, e lo collega ai due widget che sono già presenti nel mashup; infine esegue il mashup (figure 7.19, 7.20 e 7.21).

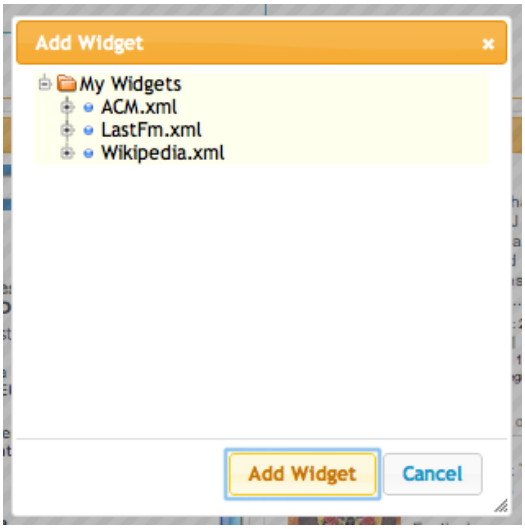

Figura 7.19: Caricamento del widget di wikipedia dalla libreria

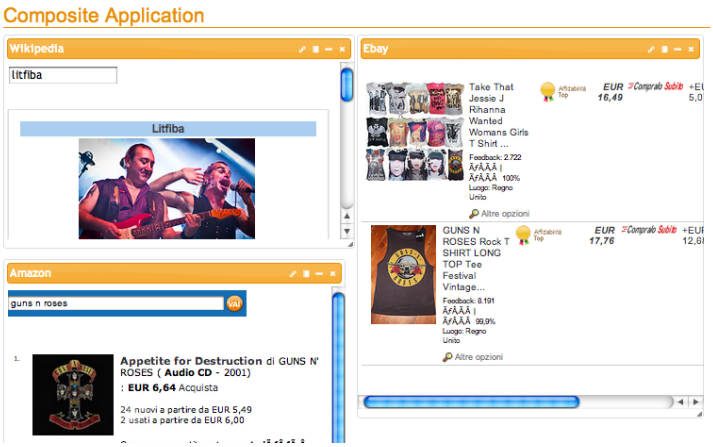

Figura 7.20: Il widget caricato dalla libreria viene inserito all'interno del mashup

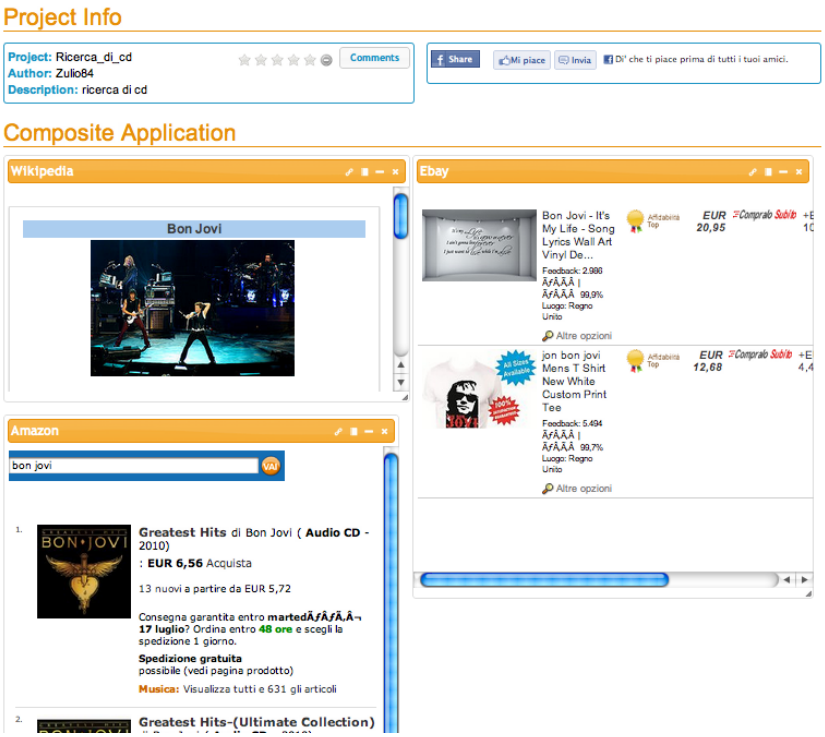

Figura 7.21: Esecuzione del mashup

Alessandra condivide le modifiche effettuate all'interno del repository (figura 7.22).

Attraverso il plugin *Send Button* invia un messaggio a Giuliano comunicando la modifica del mashup (figura 7.23).

Giuliano riceve il messaggio all'interno del social network (figura 7.24).

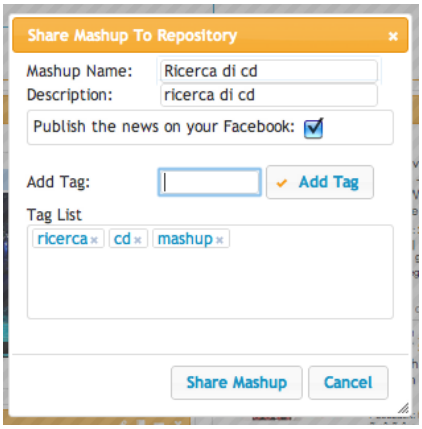

Figura 7.22: Condivisione delle modifiche nel repository

Figura 7.23: Alessandra invia un messaggio a Giuliano per informarlo della modifiche effettuate

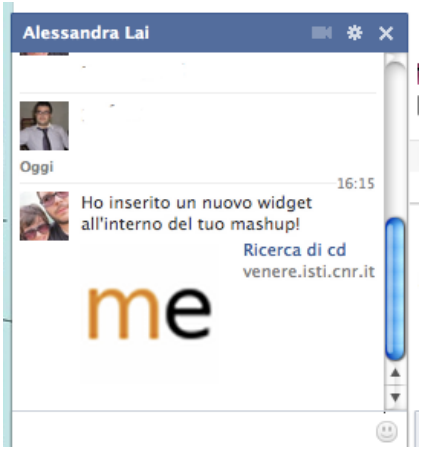

Figura 7.24: Giuliano riceve il messaggio da Alessandra

Capitolo 8

Validazione

8.1 Test di valutazione dell'usabilità

Per valutare l'usabilità della piattaforma di mashup è stato proposto un test utente suddiviso in vari task.

8.1.1 Organizzazione del test

Tramite il test utente si è voluto valutare il livello di usabilità del MashupEditor dopo le modifiche effettuate durante il periodo di tesi e, si è cercato di capire se l'ambiente grafico fosse in grado di guidare ed allo stesso tempo, facilitare l'utente nella creazione, composizione ed esecuzione di un mashup.
Prima di iniziare il test ad ogni partecipante sono state presentate le principali caratteristiche della piattaforma di mashup ed i comandi per poterle utilizzare attraverso una guida cartacea dove venivano anche descritti lo scenario del test e i task da eseguire.
Ad ogni utente sono state poste alcune domande a carattere personale, alla fine di ogni task sono state poste delle domande sulle operazioni eseguite all'interno del task, infine è stato chiesto agli utenti di rispondere ad alcune domande per raccogliere le loro impressioni generali ed eventuali critiche e/o suggerimenti.
Gli utenti hanno svolto il test da soli, uno alla volta, assistiti e coordinati

dal supervisore del test, che ha svolto anche il ruolo di utente supplementare durante il task numero 4.

8.1.2 Partecipanti

Il test è stato effettuato su un campione di 19 utenti di età compresa tra i 19 e i 30 anni (età media: 26, deviazione standard: 2,6), di cui 10 di sesso femminile e 9 di sesso maschile.
Il livello d'istruzione dei partecipanti al test è il seguente:

- 9 Diploma Scuola Superiore;
- 6 Laurea Triennale;
- 4 Laurea Magistrale o Specialistica.

Gli utenti del test hanno dichiarato di utilizzare il PC come strumento principale per la navigazione sul Web, l'utilizzo medio è di 7,21 ore al giorno, deviazione standard: 3.65 (min. 1 ora, max. 15 ore).
Il 94,7% degli utenti (18 su 19) hanno dichiarato di utilizzare almeno un Social Network, il più utilizzato dagli utenti è Facebook.

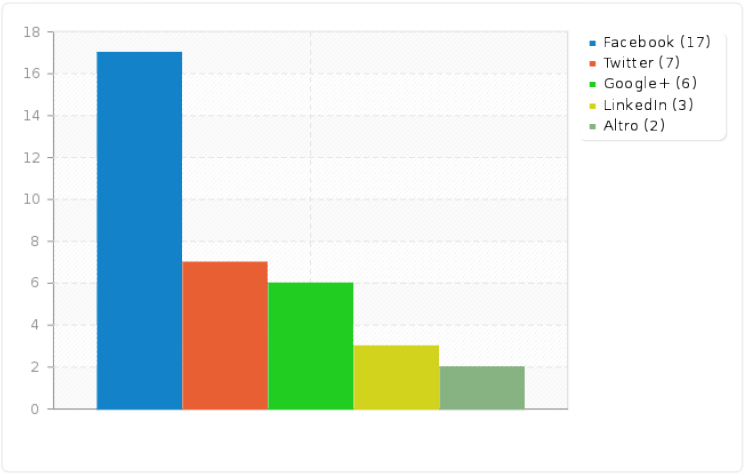

Figura 8.1: Social Network utilizzati dagli utenti del test

Gli utenti sono stati scelti per conoscenza diretta, infatti sono stati scelti 15 utenti classificabili come "*End Users*", in quanto non lavorano o studiano nel settore dell'informatica, 2 studenti iscritti alla facoltà di informatica mentre ultimi 2, lavorano come programmatori.
Infine 18 dei 19 utenti non avevano mai utilizzato strumenti per la creazione di mashup, l'unico utente che invece aveva esperienze con tool simili ha utilizzato una precedente versione della piattaforma.

8.1.3 Scenario e struttura del test

Lo scenario proposto all'utente è il seguente:

L'utente è un collezionista di CD musicali che effettua spesso acquisti online. Per scegliere il nuovo CD da acquistare utilizza "www.wikipedia.it" per cercare le informazioni sul gruppo musicale a cui è interessato, per sapere quali sono i CD che tale gruppo ha pubblicato effettua delle ricerche su "www.lastfm.it".
Infine utilizza il sito "www.amazon.it" per l'acquisto di CD nuovi mentre ricerca i CD usati su "www.ebay.it".

Il test è composto da 4 task principali, all'interno di ognuno l'utente ha utilizzato una parte delle nuove funzionalità aggiunte alla piattaforma:

- ***Task 1***: Creazione di una nuova applicazione, composta da due widget creati tramite la navigazione dal Web, creazione di una connessione tra i due widget ed esecuzione del mashup creato. Salvataggio di un widget nella propria Library personale.

  Le operazioni eseguite dagli utenti sono:

  - Effettuare il login alla piattaforma.
  - Creare un nuovo progetto ed inserire una piccola descrizione.
  - Creare un nuovo widget tramite la navigazione dal Web: collegarsi a "www.wikipedia.it", dove si ricercano informazioni su "*LITFIBA*".

  - Una volta apparsa la pagina coi risultati, selezionare il riquadro con le informazioni generali sul gruppo e inviarle alla piattaforma.
  - Spostarsi sulla piattaforma e inserire il nome del nuovo widget creato: "*Wikipedia*".
  - Si ripeta l'operazione per il sito "www.amazon.it", creando il widget "*Amazon*", in questo caso selezionare i primi due risultati della ricerca.
  - Attivare la modalità di registrazione.
  - Creare una connessione tra i due widget creati facendo il copia-incolla del contenuto della casella di ricerca del widget di "*Amazon*" all'interno di quella del widget di "*Wikipedia*".
  - Fermare la modalità di registrazione.
  - Eseguire il mashup creato cercando "*METALLICA*".
  - Salvare il widget di "*Wikipedia*" nella propria libreria personale inserendo le informazioni relative a quel widget.
  - Salvare il progetto e chiuderlo.

- ***Task 2***: Caricamento del proprio mashup, aggiunta di un nuovo widget creato da un altro utente, aggiornamento delle connessioni tra i widget, esecuzione del mashup e condivisione del progetto all'interno del repository.

  L'utente ha eseguito le seguenti operazioni:

  - Caricare il progetto creato nel Task 1.
  - Effettuare una ricerca di un widget nel repository: cercare i widget dell'autore "*Zulio84*" quindi caricare il widget di "*LastFM*".
  - Collegarlo agli altri due già presenti: eseguire il copia-incolla del contenuto della casella di ricerca del widget "*Amazon*" in quella del widget di "*Wikipedia*" e di "*LastFM*".

  - Eseguire il Mashup cercando "*LITFIBA*".
  - Condividere il progetto nel repository inserendo le informazioni sul progetto.
  - Salvare e chiudere il progetto.

- ***Task 3***: Caricamento del mashup condiviso da un altro utente, aggiunta di un widget dalla propria libreria dei widget, aggiornamento delle connessioni, esecuzione e condivisione delle modifiche effettuate.

  Elenco delle operazioni svolte dagli utenti:

  - Effettuare la ricerca nel repository dei progetti, ricercando i progetti dell'autore "*Zulio84*".
  - Caricare il progetto "*user_test_default*".
  - Caricare dalla propria libreria il widget salvato in precedenza.
  - Connetterlo agli altri widget già presenti.
  - Eseguire il mashup cercando "*BON JOVI*".
  - Valutare il progetto, quindi inserire un commento per il progetto utilizzato.
  - Condividere le modifiche effettuate.
  - Salvare e chiudere il progetto.
  - Eseguire Logout dalla piattaforma.

- ***Task 4***: Utilizzo della funzionalità di login fornita dalla piattaforma Facebook, creazione di un progetto e condivisione sul social network; caricamento ed esecuzione del progetto di un'altro utente attraverso il link pubblicato nel social network, utilizzo del plugin dei commenti di Facebook per condividere i commenti al mashup utilizzato.

  Elenco delle operazioni effettuate durante il task:

  - Effettuare il login tramite il "*Login Button*" di Facebook.

- Creare un nuovo progetto.
- Aggiungere dal repository i widget di "*Wikipedia*", "*Amazon*" e "*Ebay*", cercandoli tra quelli dell'utente "*Zulio84*".
- Connetterli tra di loro.
- Condividere il progetto nel repository e su Facebook tramite il comando di condivisione.
- Spostarsi sulla propria "*timeline*" per visualizzare le azioni che sono state pubblicate.
- Chiudere il progetto.
- Attraverso Facebook, cercare il link al progetto "*user_test_default*" pubblicato dall'utente "*Hiis Lab*" e aprirlo.
- Eseguire il mashup cercando "*BON JOVI*".
- Inserire un commento sul mashup utilizzato tramite il plugin di Facebook "*Comments*".
- Verrà mostrato all'utente, dopo aver caricato il progetto tramite l'account dell'utente del laboratorio, che il suo commento appena inserito viene visualizzato correttamente dagli altri utenti.

8.2 Analisi dei risultati del test per i task proposti

Per il test è stato chiesto agli utenti di valutare le varie funzionalità della piattaforma esprimendo un giudizio con una scala di valutazione 1-5, dove 1 è il giudizio più negativo mentre 5 è il più positivo, fornendo anche eventuali commenti.
Di seguito, per ogni domanda posta agli utenti vengono riportati il giudizio minimo e massimo, la media e la deviazione standard. Inoltre, vengono indicati, ove presenti, eventuali opinioni, suggerimenti espressi dagli utenti.
Al termine dell'analisi delle singole domande di ogni task, vengono riportate le impressioni raccolte dall'osservatore durante ogni task del test utente.

8.2.1 Task 1

1. *Come valuti il meccanismo di creazione di un nuovo widget tramite la navigazione in una nuova finestra del browser? - ([2, 5], M: 4,31, SD: 0,75)*

   Quasi tutti gli utenti hanno giudicato positivamente questo meccanismo, infatti hanno riconosciuto con facilità il comando nell'editor e sono riusciti a selezionare le parti delle applicazioni necessarie per comporre i widget, un solo utente ha avuto necessità di assistenza per compiere l'operazione.

2. *Trovi intuitivo il sistema di selezione degli elementi all'interno della pagina visitata e il sistema di invio della seleziona alla piattaforma? - ([3, 5], M: 4,37, SD: 0,76)*

   Il giudizio globale degli utenti è positivo, due utenti hanno però consigliato di modificare l'icona che avvia l'invio della selezione alla piattaforma di mashup, sostituendola con una che sia più comprensibile.

3. *Giudichi intuitivo il meccanismo di connessione tra due widget? - ([2, 5], M: 3,63, SD: 1,07)*

   Gli utenti hanno giudicato il meccanismo buono, tutti gli utenti sono riusciti ad effettuare la connessione tra i due widget, alcuni di essi però, hanno chiesto assistenza per effettuare l'operazione. Sono stati indicati dei suggerimenti per migliorare questo meccanismo:

   - Inserire l'icona per identificare il comando di registrazione, da affiancare a quello presente nel menù;
   - Evidenziare quali sono i componenti da utilizzare per le connessioni;
   - Modificare l'aspetto del tasto che ferma la registrazione, poiché rimane più in evidenza rispetto alle caselle di ricerca dei widget e l'utente è portato a cliccarci senza aver completato le connessioni.

4. *Come valuti il sistema di visualizzazione delle connessioni tra i widget? - ([3, 5], M: 4,21, SD: 0,63)*

   Gli utenti hanno trovato molto comprensibile questo sistema, apprezzando la visualizzazione delle connessioni tramite le frecce che collegano i widget.

5. *Trovi corretto il sistema di nascondere la parte di ricerca dei widget che vengono collegati e aggiornati da altri widget?- ([3, 5], M: 4,58, SD: 0,69)*

   Gli utenti si sono mostrati favorevoli a questa soluzione, infatti riuscivano a capire che il collegamento da loro creato modificava l'aspetto del widget collegato ed intuivano la dipendenza che si era creata tra i due widget, un solo utente ha consigliato di lasciar visibili anche le caselle di ricerca dei widget collegati ma disabilitando l'input da parte dell'utente.

6. *Come valuti il sistema di esecuzione del mashup?- ([3, 5], M: 4,42, SD: 0,69)*

   Gli utenti hanno trovato semplice eseguire il mashup da loro creato.

7. *Come valuti il sistema di aggiunta di un widget alla libreria personale? - ([2, 5], M: 4,37, SD: 1,01)*

   Gli utenti sono riusciti ad aggiungere il widget alla propria libreria con facilità, due utenti hanno consigliato di aumentare le dimensioni del tasto presente nella barra superiore del widget e di modificarne l'icona.

Tutti gli utenti hanno completato le operazioni indicate per il task 1, nonostante questo essi hanno trovato qualche difficoltà a causa del numero di azioni da compiere durante il task stesso. In particolare possiamo suddividere il task in due fasi, la fase di creazione dei widget tramite la selezione diretta delle componenti dell'applicazione Web e la fase di connessione ed esecuzione all'interno del MashupEditor.
Durante l'esecuzione delle operazioni della fase 1 gli utenti hanno avuto a che fare con il sistema per la selezione delle componenti di un'applicazione Web, alcuni di essi hanno avuto necessità di essere assistiti durante la fase di selezione poichè per tutti gli utenti era la prima volta che utilizzavano degli strumenti del genere. Ad esempio, alcuni utenti quando dovevano selezionare le componenti, invece di effettuare la selezione tramite il click del tasto sinistro del mouse, provavano ad effettuare la selezione col tasto destro oppure facendo il drag 'n drop delle parti che volevano selezionare.
Durante la fase due, quando bisognava effettuare la connessione tra i widget tramite il *copia-incolla*, più della metà degli utenti hanno avuto difficoltà ad avviare la modalità di registrazione e poi a capire *cosa copiare*, *da dove copiare*, e *dove incollare*. Due utenti hanno dimenticato di fermare la registrazione, continuando ad utilizzare la piattaforma in modalità di registrazione, in questo caso è stato necessario ripetere l'operazione di connessione.

8.2.2 Task 2

1. *Trovi intuitivo il sistema per il caricamento dei tuoi progetti? - ([3, 5], M: 4,32, SD: 0,67)*

   Gli utenti sono riusciti ad individuare e utilizzare con facilità il comando per caricare i progetti.

2. *Vorresti trovare più informazioni all'interno della finestra del caricamento? - (Si 15,8%, No 84,2%)*

   Gli utenti che vorrebbero maggiori informazioni hanno suggerito di inserire le anteprime dei progetti disponibili e le relative descrizioni all'interno della finestra di caricamento dei progetti.

3. *Come hai trovato il sistema per il caricamento di un widget dal repository? - ([3, 5], M: 4,37, SD: 0,76)*

   Gli utenti sono riusciti a completare l'operazione di caricamento facilmente, la maggior parte ha individuato autonomamente il comando, due utenti invece hanno utilizzato la guida al test per avere informazioni su come individuare il comando.

4. *Vorresti trovare più informazioni all'interno della finestra di caricamento? - (Si 15,8%, No 84,2%)*

   Gli utenti che hanno indicato di volere più informazioni hanno suggerito di aggiungere l'anteprima e la descrizione dei widget che si possono caricare.

5. *Trovi intuitivo il meccanismo per la condivisione di un progetto nel repository? - ([3, 5], M: 4,53, SD: 0,61)*

   Gli utenti hanno completato quest'operazione con facilità e soddisfazione, non sono stati indicati suggerimenti e consigli.

L'esecuzione del task 2 è stata portata a termine da tutti gli utenti, le operazioni specifiche sulle nuove funzionalità sono state eseguite agevolmente dagli

utenti.
All'interno del task veniva chiesto di effettuare una seconda volta la connessione tra i widget e l'esecuzione del mashup. Durante questa fase è stato osservato che circa la metà degli utenti ricordavano come avevano fatto per connettere due widget durante il task 1, mentre non ricordavano, o forse davano per scontato, di attivare la modalità di registrazione delle azioni. L'osservatore è dovuto intervenire ricordando a diversi di loro di attivare tale funzione per eseguire poi la connessione tra i widget.

8.2.3 Task 3

1. *Trovi intuitivo il meccanismo di ricerca e caricamento di un progetto dal repository? - ([1, 5], M: 4,05, SD: 1,13)*

   Gli utenti sono riusciti ad eseguire l'operazione richiesta indicando un giudizio medio abbastanza elevato. Tre utenti hanno consigliato di modificare l'etichetta del menù "*Share*" suggerendo che venga sostituita da una label "*Community*", "*Repository*" oppure "*Search & Share*".

2. *Vorresti trovare più informazioni all'interno della finestra di ricerca? - (Si 21%, No 79%)*

   Gli utenti che vorrebbero trovare più informazioni hanno suggerito di aggiungere la descrizione e l'anteprima del mashup, oltre alle indicazioni per l'esecuzione dello stesso.

3. *Trovi intuitivo il meccanismo di caricamento di un widget dalla tua libreria personale? - ([2, 5], M: 4,37, SD: 0,83)*

   Due utenti hanno suggerito di introdurre una sezione della piattaforma dedicata alla libreria dei widget.

4. *Trovi corretto poter modificare i progetti che puoi caricare dal repository? Puoi indicare il perché? - (Si 94,7%, No 5,3%)*

   Gli utenti hanno indicato le seguenti motivazioni:

- Personalizzare le applicazioni degli altri utenti secondo le proprie esigenze;
- Risparmiare tempo, evitando di ricostruire applicazioni già assemblate da altri utenti.

5. *Trovi corretto che tu possa ri-condividere un progetto che hai modificato? - (Si 94,7%, No 5,3%)*

   La tecnica adottata dalla piattaforma è stata giudicata positivamente dal 94,7% degli utenti.

6. *Pensi che chi condivide un progetto, debba dare l'opportunità di modificarlo liberamente agli altri utenti? Indicare il perché. - (Si 89,5%, No 11,5%)*

   Gli utenti che si sono mostrati favorevoli a questo sistema, hanno dichiarato che è importante, per gli utenti che utilizzano progetti creati da altri, poterli modificare a proprio piacere.
   Per contro gli utenti non favorevoli alla soluzione implementata hanno suggerito di introdurre il controllo degli utenti che possono visualizzare, caricare e modificare i mashup condivisi, dando la possibilità a chi condivide un mashup di scegliere quali utenti possono cercare, caricare e/o modificare i suoi progetti e salvaguardare la paternità del mashup.

7. *Trovi utile il meccanismo di voto e commento delle applicazioni?- ([3, 5], M: 4,63, SD: 0,68)*

   Il meccanismo di voto e commento dei mashup è stato trovato molto utile dagli utenti, non sono stati rilasciati consigli o suggerimenti.

In questo task alcuni utenti hanno avuto qualche difficoltà a trovare il comando di ricerca dei mashup nel repository, l'osservatore è intervenuto più volte per indicare la posizione del comando all'interno dell'editor.
Anche in questo task gli utenti dovevano eseguire la connessione e l'esecuzione di un mashup, il numero degli utenti che hanno completato autonomamente

quest'operazioni è aumentato rispetto al task 2, sono stati solo 5 gli utenti a cui l'osservatore ha ricordato di avviare la modalità di registrazione.

8.2.4 Task 4

1. *Trovi utile la possibilità di effettuare il login attraverso il tuo account di Facebook? - ([4, 5], M: 4,74, SD: 0,45)*

   Il giudizio medio per questa funzionalità è molto elevato. Non sono stati rilasciati suggerimenti per questa domanda.

2. *Dopo aver condiviso il tuo progetto su Facebook, analizzando la tua timeline, trovi chiaro l'aspetto delle notizie relative al Mashup Editor? - ([1, 5], M: 4,47, SD: 0,96)*

   Gli utenti hanno compreso e riconosciuto con facilità le notizie pubblicate sulla propria timeline da parte dell'applicazione, un solo utente ha avuto problemi dovuti al fatto che egli non ha attivo il "*diario*" per il proprio profilo ed ha bloccato le notifiche delle applicazioni di Facebook, per cui l'applicazione MashupEditor non ha potuto pubblicare sulla sua bacheca. All'utente è stato mostrato cosa viene pubblicato attraverso il profilo Facebook del laboratorio ed ha riconosciuto ed individuato le notizie pubblicate dall'applicazione MashupEditor.

3. *Trovi utile la possibilità di caricare i progetti cliccando sui link condivisi dagli altri utenti su Facebook? - ([4, 5], M: 4,84, SD: 0,37)*

   Gli utenti hanno accolto favorevolmente questa funzionalità della piattaforma, non sono stati rilasciati commenti o suggerimenti per migliorare questa funzione.

4. *Trovi utile la possibilità di inserire commenti ai progetti condivisi attraverso il plugin Comments di Facebook? - ([4, 5], M: 4,79, SD: 0,42)*

Gli utenti hanno trovato molto utile questa funzionalità, ed erano soddisfatti quando vedevano il proprio commento apparire all'interno del progetto aperto da un'altro utente, non sono stati rilasciati commenti o suggerimenti.

5. *Trovi che l'integrazione con i social network possa favorire il processo di creazione e condivisione delle applicazioni create dagli utenti? Puoi scrivere il perché? - (Si 100%, No 0%)*

   Gli utenti hanno espresso un giudizio unanime, sono state indicate le seguenti motivazioni:

   - I social network sono uno strumento importante nella società attuale ed è facile utilizzarli per diffondere le proprie creazioni velocemente;
   - I social network consentono di scambiarsi pareri ed opinioni, consigli e critiche sulle applicazioni condivise;
   - I social network consentono di trovare nuove idee da cui partire per sviluppare i propri mashup;
   - I social network hanno un bacino di utenti molto vasto, è facile raggiungere un grande numero di utenti.

6. *Vorresti avere la possibilità di utilizzare anche altri social network? Quali? - (Si 68,4%, No 31,6%)*

   I social network che gli utenti vorrebbero utilizzare insieme al Mashup Editor sono Twitter, Google+, Foursquare, LinkedIn e Thumblr.

Il task 4 presentava agli utenti le operazioni di connessione ed esecuzione di mashup, anche in questo caso gli utenti che hanno avuto necessità di assistenza sono diminuiti rispetto al Task 3, infatti sono stati solo 3 di essi a non ricordare come si connettono due widget. È probabile che tali utenti abbiano necessità di utilizzare la piattaforma un numero di volte superiore a quattro per apprendere il meccanismo delle connessioni.

Gli utenti hanno espresso delle buone impressioni durante l'utilizzo delle funzionalità messe a disposizione dai plugin di Facebook, questo è dovuto alla familiarità che essi hanno con l'utilizzo del social network.
Quando gli utenti hanno analizzato le notizie relative al mashup sulla propria timeline hanno individuato con facilità quali parti della notizia era stata generata tramite le informazioni che essi avevano inserito all'interno della piattaforma.
Infine la possibilità di visualizzare e rilasciare i commenti sui mashup tramite il plugin *Comments*, é stata molto apprezzata dagli utenti del test, molti di essi infatti hanno dichiarato di utilizzare in maniera assidua la funzionalità dei commenti sul social network per interagire con i propri contatti.

8.3 Considerazioni finali sui risultati del test di usabilità

In questo capitolo sono stati presentati in modo dettagliato i risultati ottenuti dal test di valutazione della piattaforma di mashup. Nel complesso l'approccio proposto nell'editor grafico è risultato piuttosto soddisfacente. La sua facilità di utilizzo si è dimostrata abbastanza alta. Questo è testimoniato anche dai tempi medi di esecuzione dei vari task, infatti in tutti i task proposti l'utente doveva eseguire la connessione di più widget e lanciare l'esecuzione del mashup composto.
Tramite l'osservazione degli utenti si è notato che essi apprendevano con facilità la metafora della connessione tra i widget tramite la registrazione delle proprie azioni e che i tempi medi dei vari task sono decrescenti (il task 4 ha un tempo medio superiore, ma in realtà l'utente effettuava una volta l'operazione di connessione e due quella di esecuzione).
Per completare il test gli utenti hanno impiegato circa un'ora ciascuno (considerando introduzione al test, esecuzione dei task e questionario).
Di seguito troviamo i tempi di esecuzione effettivi per ogni task ed il tempo totale impiegato per eseguire i quattro task, riportando il tempo minimo, quello massimo, la media e la deviazione standard (espressi in minuti):

- *Task 1*: [Min: 9, Max: 18], Media:13,84, SD: 2,24;
- *Task 2*: [3, 10], M: 6,74, SD: 1,68;
- *Task 3*: [3, 8], M: 5,26, SD: 1,16;
- *Task 4*: [5, 14], M: 8,68, SD: 2,05;
- *Totale*: [22, 43], M: 34,54 , SD: 4,95.

Per concludere possiamo dire che gli utenti:

- sono riusciti a comprendere il funzionamento e ad utilizzare la piattaforma via via con maggiore confidenza, indicando anche degli esempi di applicazioni per i quali la utilizzerebbero;
- hanno trovato abbastanza intuitivi i menù per controllare le funzionalità dell'ambiente grafico, si potrebbe migliorare l'aspetto dell'ambiente inserendo una barra dei comandi con delle icone per identificare i vari comandi. Bisognerebbe migliorare l'aspetto dei comandi per il controllo dei widget, visualizzando l'elenco delle azioni possibili come menù a comparsa;
- si sono mostrati soddisfatti del modello di esecuzione adottato, in particolare poter avere tutti i risultati aggiornati all'interno dei vari widget facendo semplicemente una ricerca. Si potrebbe aggiungere un wizard per evidenziare i passi da compiere per creare ed eseguire un mashup, da mostrare agli utenti che utilizzano la piattaforma le prime volte;
- hanno apprezzato l'integrazione con Facebook e la possibilità di potere accedere alla piattaforma selezionando i link pubblicati da altri utenti. La familiarità degli utenti con il social network fa sì che gli utenti trovino l'ambiente grafico contenente i plugin di Facebook più familiare rispetto alla versione senza plugin.

Capitolo 9

Conclusioni

Nel corso della tesi è stata mostrata la progettazione di un ambiente grafico per il mashup di applicazioni Web.
L'ambiente è composto da due componenti principali: un'ambiente grafico dove un utente può comporre i propri mashup ed un Proxy/Mashup Server di supporto per la gestione dei mashup creati.
L'ambiente grafico è classificabile come EUD Environment in quanto è stato progettato e realizzato per fornire i meccanismi per la creazione e gestione di mashup da parte degli "*end users*", utenti che non necessariamente hanno conoscenze di programmazione. Le funzionalità presenti nell'editor sono state studiate in modo da permettere all'utente un lavoro semplice e intuitivo. Oltre alla creazione di un mashup, la piattaforma consente all'utente di salvare e ricaricare i propri mashup, salvare e ricaricare i widget che li compongono.
Gli utenti della piattaforma possono ora condividere i propri mashup e widget con gli altri utenti, attraverso l'utilizzo di un repository condiviso. È stata inoltre creata una Community tra gli utenti della piattaforma per rendere possibile valutare e commentare le soluzioni degli altri utenti.
Il Proxy/Mashup Server è stato modificato per permettere le suddette operazioni: è stato implementato il supporto per consentire l'utilizzo dell'ambiente grafico da più utenti contemporaneamente ed il supporto per la Community.

Infine è stata introdotta l'integrazione con l'ambiente dei social network, in particolare con la piattaforma Facebook. Sono stati inserite all'interno dell'ambiente grafico alcune delle funzionalità più famose ed utilizzate nel social network, come i plugin *Like Button* e *Comments*.

Al termine dello sviluppo delle funzionalità descritte finora è stato effettuato un test utente. Dall'analisi dei risultati di quest'ultimo sono stati individuati i possibili sviluppi futuri:

- Raffinamento delle funzionalità esistenti nell'ambiente grafico. Tra le più importanti si segnalano:
  - modifica della veste grafica dell'editor di modo da raggruppare i comandi e le funzionalità disponibili, in maniera simile a quanto disponibile nelle applicazioni mobili;
  - introdurre l'anteprima dei widget e dei mashup che si vogliono caricare;
  - aggiungere un meccanismo che visualizzi le azioni che l'utente ha effettuato durante la fase di registrazione, da affiancare a quello della visualizzazione delle connessioni;
  - inserire la possibilità di selezionare altri tipi di elementi di input all'interno delle applicazioni Web (check box, drop down list, ecc..) e di utilizzarli all'interno dei widget;
  - progettare un sistema per potenziare le connessioni tra i widget, dando la possibilità all'utente di utilizzare tali elementi di input per realizzare delle connessioni. Fondamentale da questo punto di vista sarà avere un paradigma che sia comprensibile e facile da imparare per gli end user;
- Integrare la piattaforma di mashup con gli altri social network com *Twitter, Google+* e *LinkedIn*, per poter raggiungere il maggior numero possibile di utenti ed aumentare la dimensione della community;

- Introduzione di categorie di utenti: dare la possibilità a colui che condivide un mashup di scegliere quali tra loro possano cercare, utilizzare e/o modificare i suoi mashup;
- Creazione di una versione mobile della piattaforma.

Appendice A

Esempio di file di salvataggio XML di un mashup

A.1 Schema XSD per il salvataggio dei mashup

In questa sezione viene riportata lo schema XSD che definisce il formato dei file XML per il salvataggio dei mashup e dei widget.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns="http://giove.isti.cnr.it/MashupEnvironment"
elementFormDefault="qualified" targetNamespace="http://giove.
    isti.cnr.it/MashupEnvironment"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- VERSION 1.1 Mashup Environment Schema -->
  <!-- AUTHORS: Giuliano Pintori -->
  <!-- definition of simple types -->

    <xs:complexType name="parameter_descriptor">
        <xs:annotation>
            <xs:documentation>
                Parameter Descriptor
            </xs:documentation>
        </xs:annotation>
        <xs:attribute name="key" type="xs:string" use="required"
            />
```

```
        <xs:attribute name="name" type="xs:string" use="required
            " />
        <xs:attribute name="value" type="xs:string" use="
            required" />
        <xs:attribute name="label" type="xs:string" use="
            required" />
        <xs:attribute name="type" type="xs:string" use="required
            " />
        <xs:attribute name="checked" type="xs:boolean" use="
            optional" default="false"/>
    </xs:complexType>

    <xs:complexType name="send_input_descriptor">
        <xs:annotation>
            <xs:documentation>
                Send Input Descriptor
            </xs:documentation>
        </xs:annotation>
        <xs:attribute name="mashupDescriptorId" type="xs:string"
             use="required"/>
        <xs:attribute name="inputKey" type="xs:string" use="
            required" />
        <xs:attribute name="outputKey" type="xs:string" use="
            required"/>
    </xs:complexType>

    <xs:complexType name="entry">
        <xs:annotation>
            <xs:documentation>
                Hash Map Entry
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="value" type="parameter_descriptor"
                 minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="key" type="xs:string" use="required"
            />
    </xs:complexType>

    <xs:complexType name="hash_map">
        <xs:annotation>
            <xs:documentation>
                Hash Map
            </xs:documentation>
        </xs:annotation>
        <xs:sequence >
            <xs:element name="entries" type="entry" maxOccurs="
                unbounded" />
```

```
        </xs:sequence>
    </xs:complexType>

<!-- definition of complex types -->

    <xs:complexType name="mashup_descriptor">
        <xs:annotation>
            <xs:documentation>
                Mashup Descriptor
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:element name="tag" type="xs:string"/>
            </xs:sequence>
            <xs:sequence    minOccurs="0" maxOccurs="unbounded">
                <xs:element name="hrefs" type="xs:string"/>
            </xs:sequence>
            <xs:sequence    minOccurs="0" maxOccurs="unbounded">
                <xs:element name="sendHrefsWidgetId" type="
                    xs:string"/>
            </xs:sequence>
            <xs:sequence    minOccurs="0" maxOccurs="unbounded">
                <xs:element name="sendInputs" type="
                    send_input_descriptor"/>
            </xs:sequence>
            <xs:element name="params" type="hash_map" minOccurs=
                "0" maxOccurs="1" />
            <xs:element name="input" type="hash_map" minOccurs="
                0" maxOccurs="1" />
            <xs:element name="select" type="hash_map" minOccurs=
                "0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"
            />
        <xs:attribute name="title" type="xs:string" use="
            required"/>
        <xs:attribute name="innerHtml" type="xs:string" use="
            required" />
        <xs:attribute name="filePath" type="xs:string" use="
            required" />
        <xs:attribute name="cssPath" type="xs:string" use="
            required" />
        <xs:attribute name="queryInnerHtml" type="xs:string" use
            ="required" />
        <xs:attribute name="queryCssPath" type="xs:string" use="
            required" />
        <xs:attribute name="queryFilePath" type="xs:string" use=
            "required" />
```

```
        <xs:attribute name="width" type="xs:string" use="
            required"/>
        <xs:attribute name="height" type="xs:string" use="
            required"/>
        <xs:attribute name="head" type="xs:string" use="required
            "/>
        <xs:attribute name="originalUrl" type="xs:string" use="
            required" />
        <xs:attribute name="nextUrl" type="xs:string" use="
            required" />
        <xs:attribute name="windowId" type="xs:integer" use="
            required" />
        <xs:attribute name="author" type="xs:string" use="
            required" />
        <xs:attribute name="creationDate" type="xs:date" use="
            required"/>
        <xs:attribute name="lastModifiedDate" type="xs:date" use
            ="required" />
        <xs:attribute name="icon" type="xs:string" use="required
            " />
    </xs:complexType>

    <xs:complexType name="mashup_project_descriptor">
        <xs:annotation>
            <xs:documentation>
                Mashup Project Descriptor
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:element name="tag" type="xs:string"/>
            </xs:sequence>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:element name="elements" type="column"/>
            </xs:sequence>
        </xs:sequence>
        <xs:attribute name="p_id" type="xs:string" use="required
            " />
        <xs:attribute name="projectName" type="xs:string" use="
            required" />
        <xs:attribute name="widgetNumber" type="xs:integer" use=
            "required" />
        <xs:attribute name="columnNumber" type="xs:integer" use=
            "required" />
        <xs:attribute name="projectDirectory" type="xs:string"
            use="required" />
        <xs:attribute name="projectFilesDirectory" type="
            xs:string" use="required" />
        <xs:attribute name="author" type="xs:string" use="
```

```
            required" />
        <xs:attribute name="creationDate" type="xs:date" use="
            required"/>
        <xs:attribute name="lastModifiedDate" type="xs:date" use
            ="required" />
        <xs:attribute name="description" type="xs:string" use="
            required" />
    </xs:complexType>

    <xs:complexType name="column">
        <xs:annotation>
            <xs:documentation>
                Editor Column
            </xs:documentation>
        </xs:annotation>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="elements" type="mashup_descriptor"
                />
        </xs:sequence>
    </xs:complexType>

    <xs:element name="mashup_project_descriptor" type="
        mashup_project_descriptor"/>
</xs:schema>
```

A.2 Esempio di file XML di salvataggio di un mashup

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mashup:project xmlns:mashup="http://giove.isti.cnr.it/
    MashupEnvironment" p_id="1339686564158"
projectName="user_test_default" columnNumber="2" author="Zulio84
    "
creationDate="2012-06-14+02:00" lastModifiedDate="2012-06-14+02
    :00" description="user test project">
    <mashup:tag>user</mashup:tag>
    <mashup:tag>test</mashup:tag>
    <mashup:elements>
        <mashup:elements
  id="widget_4" title="Amazon"
  innerHtml="" filePath="1339685452449.html"
cssPath="http://venere.isti.cnr.it:8080/mig/Output/553
    B746328FED1BC5901093DB57D1099_4/Desktop/1339685452449_mod.css
    "
```

```
queryInnerHtml="" queryCssPath="http://venere.isti.cnr.it:8080
    /mig/Output/553B746328FED1BC5901093DB57D1099_4/Desktop/
    query1339685452449_mod.css"
queryFilePath="query1339685452449.html" height="100" head="100
    "
originalUrl="http://www.amazon.it/s/ref=nb_sb_noss
    /280-0563934-2004547&amp;__mk_it_IT=ÅMÕZÑÑ&amp;url=search-
    alias=aps&amp;field-keywords=aerosmith&amp;x=11&amp;y=13"
windowId="4" author="Zulio84"
creationDate="2012-06-14+02:00" lastModifiedDate="
    2012-06-14+02:00">
        <mashup:tag>Amazon</mashup:tag>
        <mashup:tag>amazon</mashup:tag>
        <mashup:tag>search</mashup:tag>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Rocks-
            Aerosmith/dp/B000026FT7/ref=sr_1_1?ie=UTF8&amp;
            qid=1339685414&amp;sr=8-1</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Rocks-
            Aerosmith/dp/B000026FT7/ref=sr_1_1?ie=UTF8&amp;
            qid=1339685414&amp;sr=8-1</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Rocks-
            Aerosmith/dp/B000026FT7/ref=sr_1_1?ie=UTF8&amp;
            qid=1339685414&amp;sr=8-1</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/gp/offer-
            listing/B000026FT7/ref=sr_1_1_olp?ie=UTF8&amp;qid
            =1339685414&amp;sr=8-1&amp;condition=new</
            mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/gp/offer-
            listing/B000026FT7/ref=sr_1_1_olp?ie=UTF8&amp;qid
            =1339685414&amp;sr=8-1&amp;condition=used</
            mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Rocks-
            Aerosmith/product-reviews/B000026FT7/ref=
            sr_1_1_cm_cr_acr_img?ie=UTF8&amp;showViewpoints=1
            </mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Rocks-
            Aerosmith/product-reviews/B000026FT7/ref=
            sr_1_1_cm_cr_acr_txt?ie=UTF8&amp;showViewpoints=1
            </mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/s/ref=
            sr_nr_seeall_1?rh=k%3Aaerosmith%2Ci%3Apopular&amp
```

```
            ;keywords=aerosmith&amp;ie=UTF8&amp;qid
            =1339685414</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Toys-Attic-
            Aerosmith/dp/B0000250PB/ref=sr_1_2?ie=UTF8&amp;
            qid=1339685414&amp;sr=8-2</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Toys-Attic-
            Aerosmith/dp/B0000250PB/ref=sr_1_2?ie=UTF8&amp;
            qid=1339685414&amp;sr=8-2</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Toys-Attic-
            Aerosmith/dp/B0000250PB/ref=sr_1_2?ie=UTF8&amp;
            qid=1339685414&amp;sr=8-2</mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/gp/offer-
            listing/B0000250PB/ref=sr_1_2_olp?ie=UTF8&amp;qid
            =1339685414&amp;sr=8-2&amp;condition=new</
            mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/gp/offer-
            listing/B0000250PB/ref=sr_1_2_olp?ie=UTF8&amp;qid
            =1339685414&amp;sr=8-2&amp;condition=used</
            mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Toys-Attic-
            Aerosmith/product-reviews/B0000250PB/ref=
            sr_1_2_cm_cr_acr_img?ie=UTF8&amp;showViewpoints=1
            </mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/Toys-Attic-
            Aerosmith/product-reviews/B0000250PB/ref=
            sr_1_2_cm_cr_acr_txt?ie=UTF8&amp;showViewpoints=1
            </mashup:hrefs>
        <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
            Proxy?migpar=0F0Thttp://www.amazon.it/s/ref=
            sr_nr_seeall_2?rh=k%3Aaerosmith%2Ci%3Apopular&amp
            ;keywords=aerosmith&amp;ie=UTF8&amp;qid
            =1339685414</mashup:hrefs>
    <mashup:sendInputs mashupDescriptorId="widget_5"
        inputKey="field-keywords" outputKey="_nkw"/>
        <mashup:params>
            <mashup:entries key="__mk_it_IT">
                <mashup:value key="__mk_it_IT" name="
                    __mk_it_IT" value="?M?Z??" label="&amp;
                    nbsp;" checked="false"/>
            </mashup:entries>
            <mashup:entries key="field-keywords">
```

```
                    <mashup:value key="field-keywords" name="
                        field-keywords" value="aerosmith" label="
                        &amp;nbsp;" checked="true"/>
                </mashup:entries>
                <mashup:entries key="y">
                    <mashup:value key="y" name="y" value="13"
                        label="&amp;nbsp;" checked="false"/>
                </mashup:entries>
                <mashup:entries key="url">
                    <mashup:value key="url" name="url" value="
                        search-alias" label="&amp;nbsp;" checked=
                        "false"/>
                </mashup:entries>
                <mashup:entries key="x">
                    <mashup:value key="x" name="x" value="11"
                        label="&amp;nbsp;" checked="false"/>
                </mashup:entries>
            </mashup:params>
            <mashup:input>
                <mashup:entries key="field-keywords">
                    <mashup:value key="field-keywords" name="
                        field-keywords" value="aerosmith" label="
                        &amp;nbsp;" type="text" checked="false"/>
                </mashup:entries>
            </mashup:input>
            <mashup:select/>
        </mashup:elements>
    </mashup:elements>
    <mashup:elements>
        <mashup:elements id="widget_5" title="Ebay" innerHtml=""
             filePath="1339685542264.html"
 cssPath="http://venere.isti.cnr.it:8080/mig/Output/553
     B746328FED1BC5901093DB57D1099_5/Desktop/1339685542264_mod.
     css" queryInnerHtml=""
 queryCssPath="http://venere.isti.cnr.it:8080/mig/Output/553
     B746328FED1BC5901093DB57D1099_5/Desktop/
     query1339685542264_mod.css" queryFilePath="
     query1339685542264.html" height="100" head="100"
 originalUrl="http://www.ebay.it/sch/i.html&amp;_from=R40&amp;
     _trksid=m570&amp;_nkw=aerosmith&amp;_sacat=See-All-
     Categories"
windowId="5" author="Zulio84" creationDate="2012-06-14+02:00"
    lastModifiedDate="2012-06-14+02:00">
            <mashup:tag>Ebay</mashup:tag>
            <mashup:tag>ebay</mashup:tag>
            <mashup:tag>search</mashup:tag>
            <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
                Proxy?migpar=0F1Thttp://www.ebay.it/itm/AEROSMITH
                -Walk-this-way-CD-live-raro-Fabbri-editori
```

```
        -/130707348643?pt=CD&amp;hash=item1e6ec3dca3</
        mashup:hrefs>
    <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
        Proxy?migpar=0F1Thttp://www.ebay.it/itm/AEROSMITH
        -Walk-this-way-CD-live-raro-Fabbri-editori
        -/130707348643?pt=CD&amp;hash=item1e6ec3dca3</
        mashup:hrefs>
    <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
        Proxy?migpar=0F1Thttp://www.ebay.it/itm/Aerosmith
        -Ready-To-Rock-World-Tour-2010-T-Shirt-New-
        Official-4-Sizes-/270902336601?pt=
        UK_Music_Music_Memorabilia_LE&amp;var=&amp;hash=
        item84ba023dd2</mashup:hrefs>
    <mashup:hrefs>http://venere.isti.cnr.it:8080/mig/
        Proxy?migpar=0F1Thttp://www.ebay.it/itm/Aerosmith
        -Ready-To-Rock-World-Tour-2010-T-Shirt-New-
        Official-4-Sizes-/270902336601?pt=
        UK_Music_Music_Memorabilia_LE&amp;var=&amp;hash=
        item84ba023dd2</mashup:hrefs>
    <mashup:params>
        <mashup:entries key="_from">
            <mashup:value key="_from" name="_from" value
                ="R40" label="&amp;nbsp;" checked="false"
                />
        </mashup:entries>
        <mashup:entries key="_trksid">
            <mashup:value key="_trksid" name="_trksid"
                value="m570" label="&amp;nbsp;" checked="
                false"/>
        </mashup:entries>
        <mashup:entries key="_sacat">
            <mashup:value key="_sacat" name="_sacat"
                value="See-All-Categories" label="&amp;
                nbsp;" checked="false"/>
        </mashup:entries>
        <mashup:entries key="_nkw">
            <mashup:value key="_nkw" name="_nkw" value="
                aerosmith" label="&amp;nbsp;" checked="
                true"/>
        </mashup:entries>
    </mashup:params>
    <mashup:input>
        <mashup:entries key="_nkw">
            <mashup:value key="_nkw" name="_nkw" value="
                aerosmith" label="&amp;nbsp;" type="text"
                 checked="false"/>
        </mashup:entries>
    </mashup:input>
    <mashup:select/>
```

```
            </mashup:elements>
        </mashup:elements>
</mashup:project>
```

Appendice B

Questionario Test Utente

Di seguito vengono riportati i gruppi di domande poste all'utente durante il test utente.

B.1 Informazioni personali

In questa sezione l'utente ha fornito alcune informazioni personali:

- Sesso;
- Età;
- Livello d'istruzione;
- Di solito utilizzi il pc per navigare in Internet?
- Quante ore al giorno navighi sul Web?
- Conosci ed utilizzi i social network? Se 'SI', Quale?

B.2 Esperienze dell'utente con l'uso di strumenti per il mashup

Le domande di questa sezione hanno riguardato le esperienze degli utenti con strumenti per la creazione di mashup:

- Prima di fare questo test, hai mai utilizzato dei tool per la creazione di Mashup di applicazioni Web? Se si, puoi indicare quali?

B.3 Domande sul Task 1

Domande relative al Task 1:

- Come valuti il meccanismo di creazione di un nuovo widget tramite la navigazione in una nuova finestra del browser?
- Trovi intuitivo il sistema di selezione degli elementi all'interno della pagina visitata e il sistema di invio della seleziona alla piattaforma?
- Giudichi intuitivo il meccanismo di connessione tra due widget?
- Come valuti il sistema di visualizzazione delle connessioni tra i widget?
- Trovi corretto il sistema di nascondere la parte di ricerca dei widget che vengono collegati e aggiornati da altri widget?
- Come valuti il sistema di esecuzione del mashup?
- Come valuti il sistema di aggiunta di un widget alla libreria personale?

B.4 Domande sul Task 2

Domande relative al Task 2:

- Trovi intuitivo il sistema per il caricamento dei tuoi progetti?

- Vorresti trovare più informazioni all'interno della finestra del caricamento?
- Come hai trovato il sistema per il caricamento di un widget dal repository?
- Vorresti trovare più informazioni all'interno della finestra di caricamento?
- Trovi intuitivo il meccanismo per la condivisione di un progetto nel repository?

B.5 Domande sul Task 3

Domande relative al Task 3:

- Trovi intuitivo il meccanismo di ricerca e caricamento di un progetto dal repository?
- Vorresti trovare più informazioni all'interno della finestra di ricerca?
- Trovi intuitivo il meccanismo di caricamento di un widget dalla tua libreria personale?
- Trovi corretto poter modificare i progetti che puoi caricare dal repository? Puoi indicare il perché?
- Trovi corretto che tu possa ri-condividere un progetto che hai modificato?
- Pensi che chi condivide un progetto, debba dare l'opportunità di modificarlo liberamente agli altri utenti? Indicare il perché.
- Trovi utile il meccanismo di voto e commento delle applicazioni?

B.6 Domande sul Task 4

Domande relative al Task 4:

- Trovi utile la possibilità di effettuare il login attraverso il tuo account di Facebook?
- Dopo aver condiviso il tuo progetto su Facebook, analizzando la tua timeline, trovi chiaro l'aspetto delle notizie relative al MashupEditor?
- Trovi utile la possibilità di caricare i progetti cliccando sui link condivisi dagli altri utenti su Facebook?
- Trovi utile la possibilità di inserire commenti ai progetti condivisi attraverso il plugin "*Comments*" di Facebook?
- Trovi che l'integrazione con i social network possa favorire il processo di creazione e condivisione delle applicazioni create dagli utenti? Puoi scrivere il perché?
- Vorresti avere la possibilità di utilizzare anche altri social network? Quali?

B.7 Domande finali

In questa sezione l'utente ha risposto a domande di carattere generale sulla piattaforma:

- Per quali tipi di applicazione Web pensi che sia utile il Mashup Editor?
- Indica quali sono gli aspetti positivi che hai trovato durante l'uso dell'applicazione.
- Indica quali sono gli aspetti negativi che hai trovato durante l'uso dell'applicazione.
- Indica se vuoi dei suggerimenti generali per migliorare la piattaforma.

Bibliografia

[1] Greasemonkey. https://addons.mozilla.org/it/firefox/addon/greasemonkey/.

[2] Igoogle. http://www.google.com/ig.

[3] Microsoft popfly. http://www.popfly.ms/.

[4] Wso2 mashup server. http://wso2.com/products/mashup-server/.

[5] Yahoo! pipes. http://pipes.yahoo.com/pipes/.

[6] Cypher Allen, Drews Clemens, Haber Eben, Kandogan Eser, Lin James, Lau Tessa, Leshed Gilly, Matthews Tara, and Wilcox Eric. Collaborative scripting for the web. In *No code required: Giving users tools to transform the Web*, pages 85–103, Burlington, MA, USA, 2010. Elsevier Inc.

[7] Hartmann Björn, Wu Leslie, Collins Kevin, and Klemmer Sott R. Programming by samples: Leveraging web sites to program their underlying services. In *No code required: Giving users tools to transform the Web*, pages 191–210, Burlington, MA, USA, 2010. Elsevier Inc.

[8] Allen Cypher. End user programming on the web. In *No code required: Giving users tools to transform the Web*, pages 3–21, Burlington, MA, USA, 2010. Elsevier Inc.

[9] Soriano Javier, Lizcano David, Cañas Miguel A., Reyes Marcos, and Hierro Juan J. Fostering innovation in a mashup-oriented enterprise

2.0 collaboration environment. In *System and Information Science Notes*, pages 62 – 69, Chengdu, China, July 2007. SIWN International Conference on Adaptive Business Systems (ICABS'2007).

[10] Fujima Jun, Lunzer Aran, Hornbaek Kasper, and Tanaka Yuzuru. Clip, connect, clone: Combining application elements to build custom interfaces for information access. In *No code required: Giving users tools to transform the Web*, pages 153–171, Burlington, MA, USA, 2010. Elsevier Inc.

[11] Chilton Lydia B., Miller Robert C., Little Greg, and Yu Chen-Hsiang. Why we customize the web. In *No code required: Giving users tools to transform the Web*, pages 23–34, Burlington, MA, USA, 2010. Elsevier Inc.

[12] Nebeling Michael, Leone Stefania, and Norrie Moira C. Crowdsourced web engineering and design. In *12th Intl. Conf. on Web Engineering (ICWE 2012)*, Berlin, Ger, July 2012.

[13] Ennals Rob. Intel mash maker. In *No code required: Giving users tools to transform the Web*, pages 173–188, Burlington, MA, USA, 2010. Elsevier Inc.

[14] Miller Robert C., Bolin Michael, Chilton Lydia B., Little Greg, Webber Mattew, and Yu Chen-Hsiang. Rewriting the web with chickenfoot. In *No code required: Giving users tools to transform the Web*, pages 39–62, Burlington, MA, USA, 2010. Elsevier Inc.

1

Acronimi

AJAX Asynchronous Javascript And XML

API Application Programming Interface

BA Business Analytics

BI Business Intelligence

CNR Consiglio Nazionale della Ricerca

CSS Cascading Style Sheets

DBMS Database management system

DOM Document Object Model

EUD End User Development

FBML Facebook Markup Language

HIIS Human Interfaces in Information Systems

HTML HyperText Markup Language

HTML5 HyperText Markup Language

ISTI Istituto di Scienza e Tecnologie dell'Informazione

JAXB Java Architecture for XML Binding

JSON JavaScript Object Notation

REST REpresentational State Transfer

RSS Really simple syndication

SOAP Simple Object Access Protocol

SaaS Software as a Service

XML Extensible Markup Language

XSD XML Schema Definition